AN IMAGE RETRIEVAL SYSTEM BASED ON A LINEAR SKELETON REPRESENTATION

By

WEN-CHEN HU

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1998

ACKNOWLEDGMENTS

Many thanks are given to my family, for their loving support and encouragement, and to Drs. Ritter, Chow, Fu, Randles, D. Wilson, and J. Wilson for their advisement of this work and associated publication efforts. Special gratitude goes to Dr. Ritter, for his patient advice and encouragement in the development of unifying theory.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract of Dissertation
Presented to the Graduate School of the University of Florida
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

AN IMAGE RETRIEVAL SYSTEM BASED ON
A LINEAR SKELETON REPRESENTATION

By

Wen-Chen Hu

August 1998

Chairman: Dr. Gerhard X. Ritter
Major Department: Computer and Information Science and Engineering

Accurate and fast retrieval and efficient storage are two principal needs in an image database. Image representation is the crucial factor in fulfilling these requirements. This research proposes an image representation, including an image data structure and a spatial knowledge representation, that will facilitate the above two demands. The proposed data structure, called a *line string*, encodes an image via a string of linear skeleton. The proposed spatial representation, called an *object-oriented line string*, provides a structural and easy way to specify the line string in a database query. The research shows that the line string exhibits advantages over other structures, but may require more storage and retrieval time.

Using the line-string representation, image retrieval becomes line-string matching, i.e., finding the *longest approximate common subsequence* (LACS) of two strings. The

LACS problem is divided into smaller parts in order to lessen its complexity. Some of these parts have been proven to be NP-hard. An approximation algorithm and an optimization neural network are constructed to find a near-optimal solution for the problem. A ratio bound of the approximation algorithm and a comparative performance of the two methods also are discussed.

To realize the advantages of the proposed research, a prototype traffic sign retrieval and browsing system is available on the World Wide Web, at http://www.cise.ufl.edu/~wenchen/idb. The experiment results show any image in the system can be precisely retrieved as long as each image has a different image string. The accuracy of image retrieval depends on the query details.

CHAPTER 1
INTRODUCTION

The core of an image database system (IDBS) consists of an image store and a relational text database. The image store caches raw images. The features of raw images are extracted and stored in the relational database according to the structure of the database. A successful IDBS has to furnish (i) accurate and fast retrieval and (ii) efficient storage. Image representation—namely, data structure and spatial representation—is crucial to achieving an image database that meets the above two demands. Various image representations have been proposed in the literature [8, 9, 10, 13, 33], but string representations perform best in storage and retrieval. String image representations soon may become the dominate type used in databases; nevertheless, they lack several requirements of databases. The following weak points are typical in string data structures:

1. String image data structures for symbolic objects portray the spatial relations of centroids or extreme points of objects, rather than the actual relative positions of objects.
2. Symbolic object representation falls short of characterizing an object's structural shape and size.
3. Some objects can be detected but cannot be recognized. In these cases, image retrieval depends on the objects' configurations, a requirement generally unachievable with symbolic objects.

1.1 A Line-String Image Representation

Like other image representations, however, string image representations fail to

satisfy fully the requirements of image databases. They particularly fall short in describing shapes, sizes, and relative positions of image objects. This paper proposes a new string image representation to facilitate image storage and retrieval. It includes a data structure, called a *line string*, and an object-oriented line string to represent spatial knowledge, called an *OOLS*.

A linear approximation algorithm converts the image skeleton into line segments. The endpoints, relative lengths, and relative angles of these straight lines are then stored in a line-string data structure. A line string maintains line segments in a string. Lines or points are described in polar coordinate systems. Each line or point in the plane is assigned coordinates (r, θ), where r is the length and θ is the measure of an angle with an initial side corresponding to the polar axis and a terminal side coinciding with the line or point. Image retrieval then becomes line-string matching.

An OOLS provides a more structural and lucid way to realize the line string in a database query. A query specifies the spatial relationship through a line string, an OOLS, a linear image, a skeletal image, or a segmented image. An OOLS also can reconstruct a linear image. Because an OOLS contains more information on configurations and relative positions of image objects than other string image representations, image reconstruction using the OOLS method generally is more accurate.

The number of spatial relations possessed by a line string is given by a theorem constructed step-by-step by other lemmas and theorems. The space needs of a line string are discussed, as are the comparisons of storage space and matching speed between line strings and other string data structures. It shows that a line string exhibits some advantages over other string image data structures but may require more storage and retrieval time.

1.2 A Line-String Matching Method

Using the line-string representation, image retrieval becomes line-string matching, i.e., finding the *longest common subsequence* (abbreviated LCS) between two images. However, a longest common subsequence does not always reveal the degree of difference between two strings required for this project. The problem is generalized to a *longest approximate common subsequence* (abbreviated LACS) problem, which is finding a maximum-gain approximate common subsequence of two strings. An approximate subsequence of a string X is a string edited from a subsequence of X. String Z is an approximate common subsequence of two strings X and Y if Z is an approximate subsequence of both X and Y. The gain function g assigns a nonnegative real number to each subsequence. The problem is divided into smaller parts in order to lessen its complexity. Some of these parts have been proven to be NP-hard. An approximation algorithm and an optimization neural network are constructed to find a near-optimal solution for the problem.

An approximation algorithm finds an approximate LACS (abbreviated ALACS) of two strings. The procedure APPROX_LACS repeatedly calls an LCS-routine, beginning with allowing zero number of edit operations for each selected symbol. Each round, the allowable number of edit operations is increased by one. The symbols selected are marked off from the input strings to prevent consideration in the next round. A ratio bound of the approximation algorithm is also discussed. A modified Hopfield neural network is designed to solve the LACS problem. A technique of interception is used to determine the values of network coefficients. The comparative performance of the two methods and some possible LACS applications are discussed.

1.3 An Image Retrieval System

To realize the advantages of the proposed research, a prototype traffic sign retrieval and browsing system is available on the World Wide Web, at http://www.cise.ufl.edu/~wenchen/idb. The system interface provides the following choices:

1. Polar or rectangular coordinate system;
2. LCS, LACS, ALACS, or Hopfield neural network search method; and
3. Centroid, endpoint, or line string.

The length and angle data, given by the polar system, and the x and y coordinates data, given by the rectangular system, supply different and useful information. It is an advantage for the system to provide both coordinate systems. The experiment results show the ALACS method outperforms the LCS method. Any image in the system can be precisely retrieved as long as each image has a different image string. The accuracy of image retrieval depends on the query details.

CHAPTER 2
RELATED RESEARCH

The next generation of active image information systems should be designed based upon the notions of generalized icons and active indexes, resulting in smart images.[7] A generalized icon is an object consisting of a pair made of the image (physical part) and its logical interpretation (logical part). If image indexing methods are active, partial, dynamic, visible, and imprecise, they are active indexes. To reach the above notions is the goal of the proposed image system. The line-string image representation, which consists of a pair of an image and its image strings, is a generalized icon. The LACS search method has not fully met the characteristics of active indexes yet.

2.1 Related String Image Representations

Chang and his colleagues [10, 11] suggested a *2-D string* that maintains the objects' spatial knowledge of their location within images. Images are stored through logical pictures retained in 2-D strings. A retrieval query may specify a 2-D string, transforming retrieval into a 2-D subsequence matching. Figure 1 exhibits a symbolized image *f*, with

	b	
		c
a	a	

Figure 1. A symbolic image f for 2-D string.

each symbol corresponding to an object. We shall briefly discuss how to create the 2-D

string from an image. Let $V = \{O_1, O_2, \ldots, O_n\}$ be a set of symbols, where each O_i, $i = 1, 2, \ldots, n$ represents a pictorial object. Let A = {'=', '<', ':'} denote a set of relations, where each element in A represents the spatial relation between two objects. Here '=' represents the spatial relation "at the same x or y location as," '<' represents the spatial relation "to the west of or the south of," and ':' represents the relation "in the same set as." A 2-D string over $V \cup A$ is represented as $(O_1\ r_1^x\ O_2 \ldots r_{n-1}^x\ O_n,\ O_{p(1)}\ r_1^y\ O_{p(2)} \ldots r_{n-1}^y\ O_{p(n)})$, where $r_1^x, r_2^x, \ldots, r_{n-1}^x, r_1^y, r_2^y, \ldots, r_{n-1}^y$ are relations from the set A and p is a permutation function on $\{1, 2, \ldots, n\}$. For instance, the 2-D string of Figure 1 is given by $(a < a = b < c,\ a = a < c < b)$. Lee *et al.* [36] defined a similarity measure based on a 2-D string's LCS. The algorithm for similarity retrieval is to transform the 2-D string LCS problem to the maximal complete subgraph (clique) problem. Lee and Shan [35] proposed another efficient methods for retrieval by objects, retrieval by pairwise spatial relationships and retrieval by subpicture. All of the methods are based on the superimposed coding technique.

Sabharwal and Bhatia [4] proposed a method for converting a 2-D strings into triples. Each triple is mapped to a unique hash address for timely retrieval of pictures, reducing the pattern-matching problem corresponding to a query to that of computation of a hash function. The values associated with the picture objects are used to compute hash addresses for triples developed from the query. Heuristics are proposed to speed up the computation of the associated values for the picture objects. Experimental results show that the new algorithm achieves almost a 90% gain, in search space, over existing algorithms to compute the associated values. They also presented a perfect hash table algorithm for image retrieval 2-D strings [42]. In that algorithm, an additional heuristic leading to a *90%* reduction in search space over the earlier algorithm. The new heuristic promises to generate a minimal perfect hash function for each experimental data set,

which was not possible with the earlier algorithm. Every time the database is modified, the hash table needs to be recomputed. This limits the use of the hash table to applications where the image database is relatively fixed. They later presented an enhancement of the perfect hash table to allow for insertion and deletion in the hash table.[43] The new hash table allows for a relatively small number of collisions and is called *near-perfect hash table*. It provides the flexibility of a live database while closely approximating the efficiency of retrieval with the perfect hash table.

Chang and Lee [9] defined an ordered triple representation that transforms each image or query into a set of ordered triples (O_i, O_j, r_{ij}), where O_i and O_j are two symbolic objects and r_{ij} denotes the spatial relationship between O_i and O_j. They then construct a hashing table for all (O_i, O_j, r_{ij}) of all images in the image database. By searching the preconstructed hashing table for all of the (O_i, O_j, r_{ij}) associated with a query, the image satisfying that query is easily determined. Wu and Chang [52] applied the geometric hashing scheme to retrieve the most similar picture from an iconic image database. They transform each iconic picture or query into a set of ordered triples (O_i, O_j, R_{ij})'s, where O_i and O_j are two iconic objects and R_{ij} is the spatial relationship code between O_i and O_j. They then use a geometric hashing scheme to construct a set of hashing functions corresponding to predefined relationship codes. Associated with these hashing functions, the most similar iconic pictures satisfying a specified query can be determined.

A modified image data structure based upon the 2-D string, called the *modified 2D string*, was introduced by Chen and Chang [13]. A similarity retrieval algorithm for the modified 2D string, also derived from the 2-D string LCS algorithm, proves more efficient than the previous algorithms.

A *2D C-string* for spatial knowledge representation that employs a cutting mechanism and a set of spatial operators was proposed by Lee and Hsu [33]. They extended

2-D strings to represent more types of spatial relationships between image objects and to provide for more efficient representation and manipulation of images. Since a spatial relationship is a fuzzy concept, the capability of similarity retrieval is essential in image database systems. They also suggested a spatial query and spatial reasoning based on a 2D C-string representation [34]. Huang and Jean [24] proposed a *2D C^+-string* representation scheme which extended the work of Lee and Hsu [33] of 2D C-string by including relative metric information about the picture into the strings. Such information can be easily obtained from a preprocessor by counting the pixels of an image. The 2D C^+-strings offer the following major advantages: (1) more accuracy in picture reconstruction; (2) less ambiguity in similarity retrieval; (3) reasoning about relative sizes, locations, and distances for a symbolic picture is possible.

A rotation-invariant spatial knowledge representation called *RS-string* was proposed by Huang and Jean [25]. The RS-string has the same representation structure as the 2D C-string. However, the semantics and conditions of the spatial operators in RS-strings are defined by a polar coordinates system instead of a Cartesian coordinates system. The RS-string representation preserves the desired rotation-invariant property because it is based on a polar coordinates system.

The method of *relative coordinates oriented symbolic* (*RCOS*) strings [8] uses an object-oriented structure that avoids partitioning. This method applies the concept of rank in simplifying data representation.

Definition I. The rank of each object symbol in a 1-D string is defined as the position of the object symbol minus the number of symbol '=' proceeding this object symbol in the string.

Let x_{i1} and x_{i2} denote the beginning rank and the end rank along the x-axis, respectively,

and let y_{i1} and y_{i2} denote the beginning rank and the end rank along the y-axis, respectively. The tuples (x_{i1}, y_{i1}) and (x_{i2}, y_{i2}) stand for coordinates corresponding to the object symbol O_i, where (x_{i1}, y_{i1}) denotes the lower-left coordinate and (x_{i2}, y_{i2}) denotes the upper-right coordinate. The symbolic object O_i is enclosed by a minimum bounding rectangle. Taking Figure 2 as an example, the corresponding RCOS string is expressed as an ordered

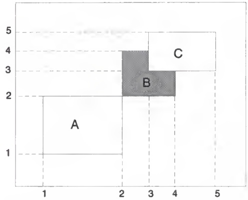


Figure 2. A symbolic picture g for RCOS string.

list [ABC, $(1,1)(2,2)$, $(2,2)(4,4)$, $(3,3)(5,5)$], where A, B, and C are three object symbols in the picture g with $(1,1)$ and $(2,2)$ denoting a pair of relative coordinates corresponding to A, with $(2,2)$ and $(4,4)$ denoting a pair of relative coordinates corresponding to B, and with $(3,3)$ and $(5,5)$ denoting a pair of relative coordinates corresponding to C.

2.2 Related String Matching Methods

Finding a *longest common subsequence* (abbreviated LCS) is mainly used to measure the discrepancies between two strings. The LCS problem is, given two strings X and Y, to find a maximum length common subsequence of X and Y. A subsequence of a given string is just the given string with some symbols (possibly none) left out. String Z is a common subsequence of X and Y if Z is a subsequence of both X and Y. Hirschberg [21] suggested two algorithms to solve the problem. The first algorithm is applicable

in the general case and requires $O(pn+nlogn)$ time where p is the length of the longest common subsequence. The second algorithm requires time bounded by $O(p(m+1-p)logn)$. In the common special case where p is close to m, this algorithm takes much less time than n^2. The LCS problem is a special case of the problem of computing edit distances. The edit distance between two character strings can be defined as the minimum cost of a sequence of editing operations which transforms one string into the other. Masek and Paterson [39] described an algorithm for computing the edit distance between two strings of length n and m, $n \geq m$, which requires $O(n \times \max(1, m/logn))$ steps whenever the costs of edit operations are integral multiples of a single positive real number and the alphabet for the strings is finite.

The *string-to-string correction* problem, first suggested by Wagner and Fischer [49], determines the distance between two strings as measured by the minimum cost sequence of edit operations required to change the first string into the other. The edit operations investigated allow insertion, deletion and change. This problem applies to keypunched words for undoing certain common keypunch errors. Lowrance and Wagner [38] proposed the *extended string-to-string* problem to include in the set of allowable edit operations the operation of interchanging the positions of two adjacent symbols. This swap operation models the error-introduction process, such as in typing, so that the search for the minimum-cost edit sequence may identify errors better. An instance of this problem, allowing only deletion and swap operations, was proven to be an *NP*-complete problem by Wagner in 1975.[48]

The *string matching* problem, given strings P and X, examines the text X for an occurrence of the pattern P as a substring, namely, whether the text X can be written as $X = YPY'$, where Y and Y' are strings. String matching is an important component of many problems, including text editing, bibliographic retrieval, and symbol manipulation.

Several algorithms [29, 31, 5] for this problem have appeared in the literature. In some instances, however, the pattern and/or the text are not exact. For example, the name may be misspelled in the text. The *approximate string matching* problem reveals all substrings in X that are close to P under some measure of closeness. The most common measure of closeness is known as the edit distance, which determines whether X contains a substring P' that resembles P in at most a certain edit distance from P to P'. The editing operation, for example, may change one symbol of a string into another, delete a symbol from a string, or insert a symbol into a string. Ukkonen [47] constructed a deterministic finite-state automaton that solves the problem. Some other approximate string-matching algorithms can be found in the related literature [51, 1, 2].

2.3 Related Image Database Systems

IIDS: IIDS (Intelligent Image Database System) [11] is based on an image data structure, 2-D strings. 2-D strings provide efficient means for iconic indexing in image database systems, and spatial reasoning. A picture query can also be specified as a 2-D string. The problem of pictorial information retrieval then becomes one of 2-D string subsequence matching. IIDS supports spatial reasoning, flexible image information retrieval, visualization, and traditional image database operations.

QBIC: The QBIC (query by image content) system allows users to find pictorial information in large image and video databases based on color, shape, texture and sketches.[17] Therefore, QBIC techniques serve as a database filter and reduce the search complexity for the user. These techniques limit the content-based features to those parameters that can be easily extracted, such as color distribution, texture, shape of a region or an object, and layout. The system offers a user a virtually unlimited set of unanticipated queries, thus allowing for general purpose applications rather than catering

to a particular application. Color- and texture-based queries are allowed for both images and objects, whereas shape-based queries are allowed only for individual objects and layout-based queries are allowed only for an entire image.

Photobook: Photobook is a set of interactive tools for browsing and searching an image database.[41] The features used for querying can be based on both text annotations and image content. The key idea behind the system is semantics-preserving image compression, which reduces images to a small set of perceptually significant coefficients. These features describe the shape and texture of the images in the database. Photobook uses multiple image features for querying general purpose image databases. The user is given the choice to select features based on the appearance, shape, and texture to browse through large databases. These features can be used in any combination and with textual features to improve the efficiency and accuracy of the retrievals.

Jain and Vailaya [26] proposed a method for trademark image database retrieval based on object shape information that would supplement traditional text-based retrieval systems. This system uses a two-stage hierarchical retrieval system: (i) a simple statistical feature-based process quickly browses through a database for a moderate number of plausible retrievals; and (ii) a deformable template matching process screens the candidate set for the best matches. The proposed hierarchical content-based image retrieval algorithm has been tested on a trademark image database containing *1100* images. The retrieval system is insensitive to variations in scale, rotation, and translation. In other words, even if a query image differs from its stored representation in the database in its orientation, position, or size, the image retrieval system is able to correctly retrieval it.

CHAPTER 3
A LINE-STRING IMAGE DATA STRUCTURE

String data structures predominate in image databases [8, 9, 10, 33, 36]; nevertheless, they lack several requirements of databases. The following weak points are typical in string data structures:

1. String image data structures for symbolic objects portray the spatial relations of centroids or extreme points of objects, rather than the actual relative positions of objects.
2. Symbolic object representation falls short of characterizing an object's structural shape and size. Although RCOS strings represent each object by its beginning boundary and end boundary, they still lack the object's overall configuration.
3. Some objects can be detected but cannot be recognized. In these cases, image retrieval depends on the objects' configurations, a requirement generally unachievable with symbolic objects.

Instead of using point symbols for object representation, line segments represent an object's skeleton. This representation captures the basic shape, size, and relative position of an object; the data structure based on this representation avoids the above mentioned problems.

Figure 3 outlines our proposed method. We first skeletonize the segmented image. A linear approximation algorithm converts the image skeleton into line segments. The endpoints, relative lengths, and relative angles of these straight lines are then stored in

a line-string data structure.

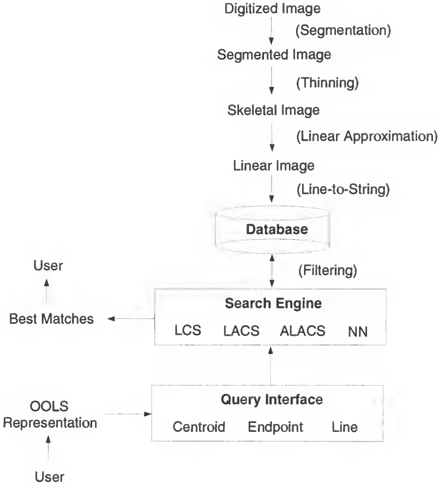


Figure 3. System structure and transformations in image representations.

3.1 A Linear Image

Zhang-Suen's method [53] is used to skeletonize a segmented image. Figure 4 shows an image and its skeleton. Next the *linear approximation* algorithm converts the skeleton into line segments. The approximation captures the essence of a skeleton in the fewest possible line segments. A polygonal approximation, based on an error function [40], is applied to this method. Before applying this algorithm, however, the skeleton has to be smoothed to unit thickness so that branch skeletons may be located. Let e be

the maximum allowable error. For a given point *A*, through which an approximating line must pass, one can define two points *B* and *C* at a distance *e* from *A*. The algorithm searches for the longest segment where the curve is contained between two parallel tangents starting from *B* and *C*.

```
APPROX_LINE (IMG, p_i, p_j, e)
        /* IMG: a skeletal image */
        /* p_i: the initial point of an object's skeleton */
        /* p_j: the neighbor of p_i at the skeleton */
        /* e: the maximum allowable error */
    b ← p_i + e
    c ← p_i - e
    IS_FIRST ← TRUE
    while NUMBER_8_NEIGHBOR(p_i) ≠ 0
        while NUMBER_8_NEIGHBOR(p_i) > 1
            p_j ← BEST_ROUTE(IMG, p_i, e)
            APPROX_LINE(IMG, p_i, p_j, e)
            print p_i
        IMG[p_i] ← CLEAR
        L̄_b ← overline(b p_j)
        L̄_c ← overline(c p_j)
        if IS_FIRST
            L̄_above ← L̄_b
            L̄_below ← L̄_c
            IS_FIRST ← FALSE
        else
```

if $\overline{L}_b$ *is above* $\overline{L}_{above}$

$\overline{L}_{above} \leftarrow \overline{L}_b$

if $\overline{L}_c$ *is below* $\overline{L}_{below}$

$\overline{L}_{below} \leftarrow \overline{L}_c$

if $\angle(\overline{L}_{above}, \overline{L}_{below}) > 0$

print p_i

$b \leftarrow p_i + e$

$c \leftarrow p_i - e$

IS_FIRST $\leftarrow$ *TRUE*

$p_i \leftarrow p_j$

$p_j \leftarrow$ *8_NEIGHBOR*(p_i)

IMG[p_i] $\leftarrow$ *CLEAR*

print p_i

The function *BEST_ROUTE* finds the route of the longest line segment when p_i has more than one 8-neighbor. This is why the skeleton needs smoothing before applying the algorithm. A non-unit thickness skeleton may mislead the algorithm into calling the *BEST_ROUTE* function. This algorithm requires quadratic time because the *BEST_ROUTE* function needs to check as many routes as possible and any point on the skeleton may invoke it. Figure 4 (c) shows the linear image that is the output of this algorithm whose input is the skeletal image of Figure 4 (b).

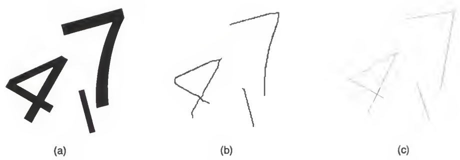


Figure 4. (a) The original image, (b) the skeletal image, and (c) the linear image.

3.2 Representation and Examples

After an image has been transformed to a linear image, an IDBS stores the image in its relational database through a data structure, called a *line string*, that maintains line segments in a string. Lines or points are described in polar coordinate systems. Each line or point in the plane is assigned coordinates (r, θ), where r is the length and θ is the measure of an angle with an initial side corresponding to the polar axis and a terminal side coinciding with the line or point.

To express the representations of line strings, let L, P, and R be three sets of symbols. Each symbol from L represents a line segment; each symbol from P represents a point or a centroid; and R = {'ϵ', '='}, where ϵ is an empty string. A *line string* written as $(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)$, r sorted by length and θ sorted by angle, is defined as

$$(c_1 \ \gamma \ c_2 \ \gamma \ \ldots \ \gamma \ c_n, \ c_{f(1)} \ \gamma \ c_{f(2)} \ \gamma \ \ldots \ \gamma \ c_{f(n)},$$

$$p_1 \ \gamma \ p_2 \ \gamma \ \ldots \ \gamma \ p_{n'}, \ p_{g(1)} \ \gamma \ p_{g(2)} \ \gamma \ \ldots \ \gamma \ p_{g(n')},$$

$$l_1 \ \gamma \ l_2 \ \gamma \ \ldots \ \gamma \ l_{n''}, \ l_{h(1)} \ \gamma \ l_{h(2)} \ \gamma \ \ldots \ \gamma \ l_{h(n'')}),$$

where

γ is over R, and γ is '=' if the ranks of the symbols on both sides are equal, otherwise γ is ϵ with rank being defined as in Chapter 2;

$c_1\ \gamma\ c_2\ \gamma\ \ldots\ \gamma\ c_n$ is a string of centroids over P with every two centroids separated by a γ and with a centroid c as the center of each object's minimum enclosed rectangle;

$p_1\ \gamma\ p_2\ \gamma\ \ldots\ \gamma\ p_{n'}$ is a string of points over P with every two points separated by a γ and with a point p as an endpoint of a line segment or an object symbol;

$l_1\ \gamma\ l_2\ \gamma\ \ldots\ \gamma\ l_{n''}$ is a string of line segments over L with every two segments separated by a γ and with a line segment l as an object's skeleton line; and

f, g, and h are permutation functions on positive integers.

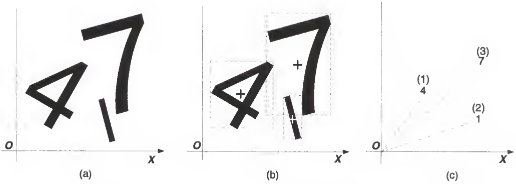


Figure 5. (a) The original image, (b) the image with centroids marked, and (c) centroids in the polar coordinate system. The number in parentheses is a length rank, and the angle ranks can be derived easily from the picture by positions.

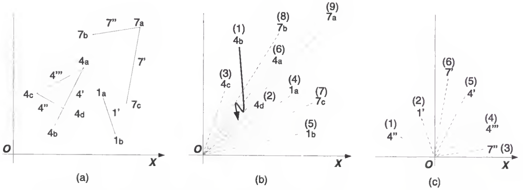


Figure 6. (a) A linear image, (b) endpoints in the polar coordinate system, and (c) linear skeleton in the polar coordinate system.

For example, the centroids in Figure 5 can be represented by a string of centroids:

$$(r_c,\ \theta_c)\ =\ (4\ 1\ 7,\ 1\ 7\ 4). \tag{2}$$

The endpoints in Figure 6 can be represented by a string of endpoints (r_p, θ_p). The elaborate form of line string, only for clarification and not to be read by computers, is shown first:

$$\begin{aligned}&\textit{elaborate form of }(r_p,\ \theta_p)\\ &= (4_b\ 4_d\ 4_c\ 1_a\ 1_b\ 4_a\ 7_c\ 7_b\ 7_a,\ 1_b\ 7_c\ 1_a\ 4_b\ 4_d\ 7_a\ 4_a\ 7_b\ 4_c),\end{aligned} \tag{3}$$

and the corresponding line string is

$$(r_p,\ \theta_p) = (4\ 4\ 4\ 1\ 1\ 4\ 7\ 7\ 7,\ 1\ 7\ 1\ 4\ 7\ 4\ 7\ 4\ 4). \tag{4}$$

The linear skeleton in Figure 6 can be represented by a string of skeleton lines(r_l, θ_l):

$$\begin{aligned}&\textit{elaborate form of }(r_l,\ \theta_l)\\ &= (4''\ 1'\ 7''\ 4'''\ 4'\ 7',\ 7''\ 4'''\ 4'\ 7'\ 1'\ 4''),\end{aligned} \tag{5}$$

and the corresponding line string is

$$(r_l,\ \theta_l) = (4\ 1\ 7\ 4\ 4\ 7,\ 7\ 4\ 4\ 7\ 1\ 4). \tag{6}$$

Combining the three line strings (2), (3), and (5) results in the line string's elaborate form for (a) of Figure 4:

$$\begin{aligned}&\textit{elaborate form of }(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)\\ &= (4\ 1\ 7,\ 1\ 7\ 4,\\ &\quad 4_b\ 4_d\ 4_c\ 1_a\ 1_b\ 4_a\ 7_c\ 7_b\ 7_a,\ 1_b\ 7_c\ 1_a\ 4_b\ 4_d\ 7_a\ 4_a\ 7_b\ 4_c,\\ &\quad 4''\ 1'\ 7''\ 4'''\ 4'\ 7',\ 7''\ 4'''\ 4'\ 7'\ 1'\ 4''),\end{aligned} \tag{7}$$

and combining the three line strings (2), (4), and (6) produces the line string

$$\begin{aligned}&(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)\\ &= (4\ 1\ 7,\ 1\ 7\ 4,\\ &\quad 4\ 4\ 4\ 1\ 1\ 4\ 7\ 7\ 7,\ 1\ 7\ 1\ 4\ 7\ 4\ 7\ 4\ 4,\\ &\quad 4\ 1\ 7\ 4\ 4\ 7,\ 7\ 4\ 4\ 7\ 1\ 4).\end{aligned} \tag{8}$$

CHAPTER 4
AN OBJECT-ORIENTED LINE-STRING SPATIAL KNOWLEDGE REPRESENTATION

Although the line-string image data structure is efficient in storage and concise in representation, users may want a more structural and clear way of describing spatial relationships in database queries. To this end, we present an object-oriented method to specify a line string. The following provides a grammar for this representation and the presents examples. The algorithms of image reconstruction and retrieval also are discussed.

4.1 Representation and Examples

The spatial knowledge representation of an image in database queries can be expressed through an OOLS. A context-free grammar for an OOLS is given by the following:

$$
\begin{array}{lcl}
OOLS & \rightarrow & \{object\ objects\} \\
object & \rightarrow & [\mathrm{id}\ centroid\ points\ lines] \\
objects & \rightarrow & object\ objects \mid \epsilon \\
centroid & \rightarrow & (\mathrm{r}_p, \theta_p) \mid \epsilon \\
points & \rightarrow & ,(\mathrm{r}_p, \theta_p)\ more\_points \mid \epsilon \\
more\_points & \rightarrow & (\mathrm{r}_p, \theta_p)\ more\_points \mid \epsilon \\
lines & \rightarrow & ,(\mathrm{p}_i, \mathrm{p}_j, \mathrm{r}_l, \theta_l)\ more\_lines \mid \epsilon \\
more\_lines & \rightarrow & (\mathrm{p}_i, \mathrm{p}_j, \mathrm{r}_l, \theta_l)\ more\_lines \mid \epsilon
\end{array}
$$

Using an OOLS, a database user has an organized way of forming an objects' spatial relationship in a query. In this grammar, the fourteen terminal symbols are

$$\mathrm{id} \quad \mathrm{r}_p \quad \theta_p \quad \mathrm{r}_l \quad \theta_l \quad \mathrm{p}_i \quad \mathrm{p}_j \quad ',' \quad \{ \quad \} \quad [\quad] \quad (\quad),$$

where id is an object symbol and where r_p and θ_p are the length and angle ranks, respectively, of a centroid or point p. Line segment $l = (\mathrm{p}_i, \mathrm{p}_j, \mathrm{r}_l, \theta_l)$ has p_i and p_j as its initial and end point sequence numbers and has r_l and θ_l as its length and angle ranks. The eight nonterminals are

$$OOLS \quad object \quad objects \quad centroid \quad points \quad more\_points \quad lines \quad more\_lines,$$

in which *OOLS* is the start symbol, *centroid* is for centroids, *points* is for endpoints, and *lines* is for line segments. Every nonterminal except *OOLS* and *object* must either be provided with a full content or with an empty string ϵ.

To clarify the semantic concept of OOLS, let O, P, and L be sets of symbols. Each symbol from O represents an object symbol. Each symbol from P is an endpoint of a line segment, an object point or the centroid of an object. A symbol from L represents a line segment. The following OOLS elliptical form is used to explain its semantic implication.

$$\begin{aligned}\{&[O_1(r_1,\theta_1),\ (r_{p_{11}},\theta_{p_{11}})\ldots(r_{p_{1x}},\theta_{p_{1x}}),\ (p_\alpha, p_\beta, r_{l_{11}}, \theta_{l_{11}})\ldots(p_\gamma, p_\delta, r_{l_{1y}}, \theta_{l_{1y}})]\ldots\\ &\left[O_i(r_i,\theta_i),\ (r_{p_{i1}},\theta_{p_{i1}})\ldots\left(r_{p_{ix'}},\theta_{p_{ix'}}\right),\ (p_{\alpha'}, p_{\beta'}, r_{l_{i1}}, \theta_{l_{i1}})\ldots\left(p_{\gamma'}, p_{\delta'}, r_{l_{iy'}}, \theta_{l_{iy'}}\right)\right]\ldots\\ &\left[O_n(r_n,\theta_n),(r_{p_{n1}},\theta_{p_{n1}})\ldots\left(r_{p_{nx''}},\theta_{p_{nx''}}\right),\ (p_{\alpha''}, p_{\beta''}, r_{l_{n1}}, \theta_{l_{n1}})\ldots\left(p_{\gamma''}, p_{\delta''}, r_{l_{ny''}}, \theta_{l_{ny''}}\right)\right]\},\end{aligned}$$

where

$O_1, \ldots, O_n$ are n object symbols over O;

(r_i, θ_i) is the representation of an object O_i's centroid over P, where r_i and θ_i are its length and angle ranks among all centroids;

$(r_{p_{ij}}, \theta_{p_{ij}})$ is the representation of an object O_i's endpoint P_{ij} over P, where $r_{p_{ij}}$ and $\theta_{p_{ij}}$ are its length and angle ranks among all endpoints;

$(p_\gamma, p_\delta, r_{l_{ij}}, \theta_{l_{ij}})$ is the representation of an object O_i's line segment l_{ij} over L, where p_γ and p_δ are its initial and end point sequence numbers and where $r_{l_{ij}}$ and $\theta_{l_{ij}}$ are its length and angle ranks among all skeleton lines. The lines always point downward and point to the right for horizontal lines.

Take the linear image at Figure 7 (c) as an example. Its centroids, endpoints, and skeleton lines in the polar coordinate systems are shown in Figure 8 (c), Figure 9 (b), and Figure 9 (c), respectively. The following steps explain how to represent the linear image by an OOLS. The OOLS' elaborate form is shown first:

$$OOLS' \ elaborate \ form \ of \ Figure \ 7 \ (c)$$

$$=\{[A(A_c),\ (a_1)(a_2)(a_3)(a_4),\ (a')(a'')(a''')]$$

$$[B(B_c),\ (b_1)(b_2)(b_3)(b_4),\ (b')(b'')(b''')]\},$$

and the OOLS is

$$OOLS \ of \ Figure \ 7 \ (\mathrm{c})$$

$$=\{[A(2,2),\ (8,7)(6,8)(1,6)(3,3),\ (1,2,4,2)(2,3,5,4)(3,4,3,6)]$$

$$[B(1,1),\ (5,5)(7,4)(4,2)(2,1),\ (1,2,2,5)(2,3,6,3)(3,4,1,1)]\}.$$

A retrieval query specified via an OOLS expression needs to be translated into a line-string query before it can be satisfied. An OOLS includes more information than its corresponding line string, e.g., an OOLS, unlike a line string, specifies the two endpoints of a line. Thus, an OOLS could easily translate into a line string. Translate an OOLS into a line string without adversely affecting system performance: The translation takes *O(N)* time for *N* symbols with each query transformed just once. Moreover, the extra space adds no cost to system memory because the IDBS stores an image's line string, rather than its OOLS.

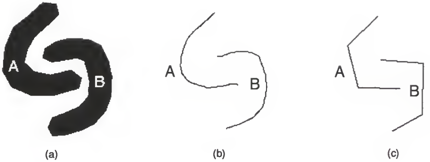


Figure 7. (a) The original image, (b) the skeletal image, and (c) the linear image.

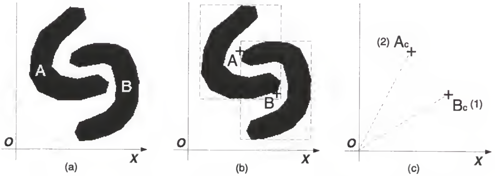


Figure 8. (a) The original image, (b) the image with centroids marked, and (c) centroids in the polar coordinate system.

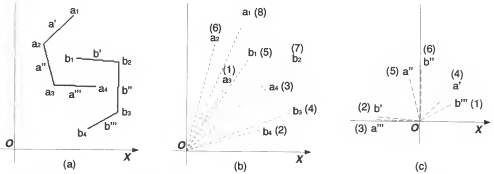


Figure 9. (a) The linear image, (b) endpoints in the polar coordinate system, and (c) linear skeleton in the polar coordinate system.

4.2 Image Restoration and Retrieval

An OOLS can reconstruct a linear image that as long as the configurations of objects are known may restore a segmented image. The reconstruction transform, which deciphers the length and angle information in an OOLS to approximate line segments, is described below. The notation used in this description arises from the OOLS grammar.

Algorithm I. Image restoration from an OOLS

Input. An OOLS spatial knowledge representation.

Output. A linear image.

begin

1. Divide the first quadrant of a polar coordinate system into $\Sigma|points|$ Chapters. Assign the point p with an ith angle rank to the ith Chapter, and set the length of p to a length d if p has a dth length rank.

2. Connect the endpoints of each line segment l according to the first two elements of the *lines* $(p_i, p_j, r_l, \theta_l)$.

3. For each line segment l, there may exist two different ranks for r_l or θ_l. One rank is from the line just created; another is from *lines* $(p_i, p_j, r_l, \theta_l)$. Adjust l then by taking the average values for r_l and θ_l.

4. Divide the first quadrant of a polar coordinate system into $\Sigma|object|$ Chapters. Assign the center of an object's skeleton line to the ith Chapter if the object's centroid has an ith angle rank, and move the center from the origin by d if the centroid has a dth length rank.

end

Figure 10 shows the reconstructed linear image of Figure 6 (a). The dotted lines show the division of a quadrant into sections.

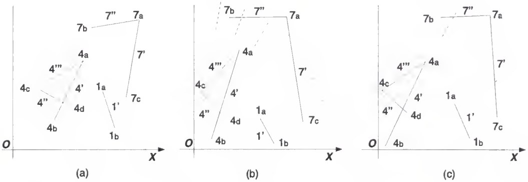


Figure 10. (a) The linear image from (a) of Figure 6, (b) the reconstructed image after the first two steps of the restoration algorithm, and (c) the final reconstructed image.

Line strings can characterize various kinds of spatial relationships described in other string image representations, but can they retrieve images correctly and effectively? The spatial relationship in an image query can be specified through any one of the following five methods: a line string, an OOLS, a linear image, a skeletal image, or a segmented image. The last four representations, however, need conversion to line strings before processing. Image retrieval then becomes line-string matching, i.e., finding a *longest approximate common subsequence* [21] between two images. Therefore, an image retrieval algorithm is needed to determine a maximum-gain approximate common subsequence of two sequences, which will be discussed later.

CHAPTER 5
ANALYSIS OF LINE STRINGS

Compared with other image data structures, line strings concisely describe elaborate spatial relationships of objects. The data structure requires relatively few symbols to create numerous line connections and distinct spatial relations. This Chapter analyzes the spatial relationships and space needs of line strings and provides comparisons of storage space and computational complexity between the method discussed and other string image data structures.

5.1 Number of Spatial Relations

Theorem V gives the number of spatial relations of a line string. It is constructed through a sequence of lemmas and theorems that establish the numbers of spatial relations of line string $(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)$ components; i.e., r_c, θ_c, r_p, θ_p, r_l, and θ_l. To establish these lemmas and theorems, we need to define the notion of *multisets*, which are sets *allowed* to contain repetitive elements.

Definition II. A multiset is like a set except that it can have repetitions of identical elements. If S and T are multisets, we define new multisets $S \uplus T$, $S \cup T$, and $S \cap T$ in the following manner: An element occurring exactly n times in S and m times in T occurs exactly $n+m$ times in $S \uplus T$, exactly max(a, b) times in $S \cup T$, and exactly min(a, b) times in $S \cap T$. [30]

For example, consider the endpoint multiset illustrated by Figure 9. This multiset is given by

$$T = \{a,\ a,\ a,\ a,\ b,\ b,\ b,\ b\}$$

and contains four a's and four b's. The nth power of a multiset T is defined next.

Definition III. If T is a multiset and n is a nonnegative integer, then

$$T^n = \begin{cases} \varnothing & if\ n = 0 \\ T^{n-1} \uplus T & otherwise \end{cases}$$

Our first lemma establishes the number of spatial relations of the centroids, i.e., r_c or θ_c in a line string. In the following, let *NC* and *NE* denote the number of centroids and the number of equal signs between these *NC* centroids, respectively.

Lemma I. For the r_c or θ_c in a line string $(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)$ with given numbers of centroids $NC>0$ and with equal signs between these centroids $0 \leq NE \leq NC-1$, the number of spatial relations NSR_c' of r_c or θ_c is given by

$$NSR_c'(NC, NE) = \left\{ {NC \atop NC-NE} \right\} (NC - NE)!$$

where "{ }" is the notation for the Stirling number of the second kind[30].

Ascertained from the above lemma, the number of spatial relations of $(r_c,\ \theta_c)$ is the product of $NSR_c'(NC, NE_r)$ and $NSR_c'(NC, NE_\theta)$.

Theorem I. For the r_c and θ_c in a line string $(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)$ with given numbers of centroids $NC>0$, equal signs NE_r between the centroids in r_c with $0 \leq NE_r \leq NC-1$, and equal signs NE_θ between the centroids in θ_c with $0 \leq NE_\theta \leq NC-1$, the number of spatial relations NSR_c of (r_c, θ_c) is given by

$$NSR_c(NC, NE_r, NE_\theta) = NSR_c'(NC, NE_r) NSR_c'(NC, NE_\theta).$$

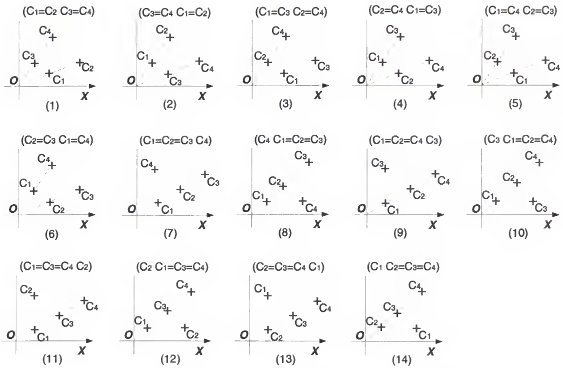


Figure 11. Fourteen spatial relations of θ_c in a line string with four centroids and two equal signs.

Consider Figure 11 as an example. In this example, we have $NSR'_c(4,2) = \left\{ {4 \atop 2} \right\} 2! = 7 \times 2! = 14$ spatial relations for θ_c in a line string with *NC=4* and *NE=2*.

Lemma II provides an equation for finding the number of spatial relations of r_p or θ_p in a line string. The collection of nonempty endpoint multisets S_h={object h's endpoints}, $1 \leq h \leq NO$ is a partition of the multiset of all endpoints U. The collection of nonempty multisets $n_i \times T_i$={endpoints separated by '='s}, $1 \leq i \leq I$ with $n_i > 0$ is a partition of multiset $\biguplus_{h=1}^{NO} S_h$. Each multiset T_i is either a single endpoint multiset or an '=' separates every two endpoints in that multiset.

Lemma II. For the r_p or θ_p in a line string $(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)$ with given numbers of objects *NO>0*, endpoints *NP≥NO*, and equal signs between these endpoints *0≤NE≤NP-1*, the number of spatial relations NSR_p' of r_p or θ_p is given by

$$NSR_p'(NO, NP, NE) = \sum_{\biguplus_{h=1}^{NO} S_h = U, |U|=NP} NSR_p''(NO, NP, NE), \ \textit{and}$$

$$NSR_p''(NO, NP, NE) = \sum_{\biguplus_{i=1}^{I} T_i^{n_i} = \biguplus_{h=1}^{NO} S_h} \frac{(NP-NE)!}{\prod_{i=1}^{I} n_i!}$$

where $\sum_{i=1}^{I} n_i = NP - NE$.

Theorem II, which is a consequence of Lemma II, provides the number of spatial relations of (r_p, θ_p). The number of spatial relations of (r_p, θ_p) is equal to

$$NSR_p(NO, NP, NE_r, NE_\theta) = \sum_{\biguplus_{h=1}^{NO} S_h = U, |U|=NP} [NSR_p''(NO, NP, NE_r) NSR_p''(NO, NP, NE_\theta) - ILL_p'(NO, NP, NE_r, NE_\theta)].$$

Among $\sum_{\biguplus_{h=1}^{NO} S_h = U, |U|=NP} NSR_p''(NO, NP, NE_r) NSR_p''(NO, NP, NE_\theta)$, the number of *illegal* spatial relations is

$$ILL_p(NO, NP, NE_r, NE_\theta) = \sum_{\biguplus_{h=1}^{NO} S_h = U, |U|=NP} ILL_p'(NO, NP, NE_r, NE_\theta).$$

$ILL_p(NO, NP, NE_r, NE_\theta) > 0$ when the statement $(|S_h \cap T_i| > \lceil |S_h|/2 \rceil) \wedge (|S_h \cap R_j| > \lceil |S_h|/2 \rceil)$, which means at least two endpoints of object h are superimposing on each other, is true. The nonempty endpoint multisets S_h={object h's endpoints}, *1≤h≤NO* are from Lemma II. The collection of nonempty multisets $n_i \times T_i$={endpoints

in r_p separated by '='s}, $1 \leq i \leq I$ with $n_i > 0$ and the collection of nonempty multisets $m_j \times R_j$={endpoints in θ_p separated by '='s}, $1 \leq j \leq J$ with $m_j > 0$ are two partitions of $\biguplus_{h=1}^{NO} S_h$.

Theorem II. For the r_p and θ_p in a line string $(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)$ with given numbers of objects $NO > 0$, endpoints $NP \geq NO$, equal signs NE_r between the endpoints in r_p with $0 \leq NE_r \leq NP-1$, and equal signs NE_θ between the endpoints in θ_p with $0 \leq NE_\theta \leq NP-1$, the number of spatial relations NSR_p of (r_p, θ_p) is given by

$$
\begin{aligned}
&NSR_p(NO, NP, NE_r, NE_\theta) \\
&\quad = \sum_{\biguplus_{h=1}^{NO} S_h = U, |U| = NP} NSR_p'''(NO, NP, NE_r, NE_\theta), \\
&NSR_p'''(NO, NP, NE_r, NE_\theta) \\
&\quad = NSR_p''(NO, NP, NE_r) NSR_p''(NO, NP, NE_\theta) - \\
&\qquad ILL_p'(NO, NP, NE_r, NE_\theta), \ and \\
&ILL_p'(NO, NP, NE_r, NE_\theta) \\
&\quad = \begin{cases} 0 & if\ P\ is\ false \\ \left[\sum_{\biguplus_{i=1}^{I} T_i^{n_i} = \biguplus_{h=1}^{NO} S_h} \frac{(NP-NE_r)!}{\prod_{i=1}^{I} n_i!} \right] \left[\sum_{\biguplus_{j=1}^{J} R_j^{m_j} = \biguplus_{h=1}^{NO} S_h} \frac{(NP-NE_\theta)!}{\prod_{j=1}^{J} m_j!} \right] & otherwise \end{cases}
\end{aligned}
$$

where statement $P = (|S_h \cap T_i| > \lceil |S_h|/2 \rceil) \wedge (|S_h \cap R_j| > \lceil |S_h|/2 \rceil)$, $\sum_{i=1}^{I} n_i = NP - NE_r$, and $\sum_{j=1}^{J} m_j = NP - NE_\theta$.

For example, the $ILL_p(2,3,1,1) = \sum_{S_h} \left(\sum_{T_i} \frac{2!}{1!1!} \right) \left(\sum_{R_j} \frac{2!}{1!1!} \right) = \sum_{S_h} \left(\frac{2!}{1!1!} \right) \left(\frac{2!}{1!1!} \right) = 4 + 4 = 8$ illegal spatial relations of a line string with $NO=2$, $NP=3$, $NE_r=1$, and $NE_\theta=1$ are shown in Figure 12.

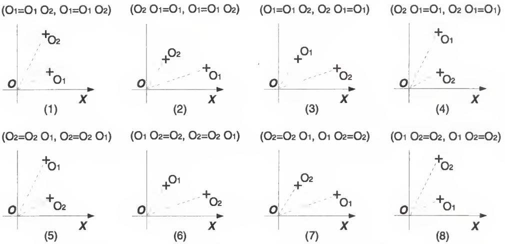


Figure 12. Eight illegal spatial relations of a line string with two objects, three endpoints, one equal sign in r_p, and one equal sign in r_θ.

Theorem III calculates the number of spatial relations of (r_l,θ_l). Lemma III, which is for constructing Theorem III, provides a formula to find the number of spatial relations of r_l or θ_l in a line string. The nonempty endpoint multisets S_h={object h's endpoints}, $1 \leq h \leq NO$ are from Lemma II. The collection of line segment multisets T_i={object i's line segments}, $1 \leq i \leq NO$ with $0 \leq |T_i| \leq \binom{|S_i|}{2}$ is a partition (allowed empty multisets) of the multiset of all line segments V. The collection of nonempty multisets $m_j \times R_j$={line segments separated by '='s}, $1 \leq j \leq J$ with $m_j > 0$ is a partition of multisets $\biguplus_{i=1}^{NO} T_i$.

Lemma III. For the r_l or θ_l in a line string $(r_c,\theta_c,r_p,\theta_p,r_l,\theta_l)$ with given numbers of objects $NO>0$, endpoints $NP \geq NO$, line segments $0<NL \leq \binom{NP-NO+1}{2}$, and equal signs between the line segments $0 \leq NE \leq NL-1$, the number of spatial relations NSR_l' of r_l or θ_l is given by

$$NSR_l'(NO,NP,NL,NE) = \sum_{\biguplus_{h=1}^{NO} S_h=U,|U|=NP} \; \sum_{\biguplus_{i=1}^{NO} T_i=V,|V|=NL} NSR_l''(NO,NL,NE), \textit{ and}$$

$$NSR_l''(NO,NL,NE) = \sum_{\biguplus_{j=1}^{J} R_j^{m_j} = \biguplus_{i=1}^{NO} T_i} \frac{(NL-NE)!}{\prod_{j=1}^{J} m_j!}$$

where $\sum_{j=1}^{J} m_j = NL - NE.$

Theorem III, which is a consequence of Lemma III, provides the number of spatial relations of (r_l, θ_l). The number of spatial relations of (r_l, θ_l) is

$$NSR_l(NO, NP, NL, NE_r, NE_\theta)$$
$$= \sum_{\biguplus_{h=1}^{NO} S_h = U, |U| = NP} \sum_{\biguplus_{i=1}^{NO} T_i = V, |V| = NL} [NSR_l''(NO, NL, NE_r)$$
$$NSR_l''(NO, NL, NE_\theta) - ILL_l'(NO, NL, NE_r, NE_\theta)].$$

Among $\sum_{\biguplus_{h=1}^{NO} S_h = U, |U| = NP} \sum_{\biguplus_{i=1}^{NO} T_i = V, |V| = NL} NSR_l''(NO, NL, NE_r) NSR_l''(NO, NL, NE_\theta)$, the number of *illegal* spatial relations is

$$ILL_l(NO, NP, NL, NE_r, NE_\theta)$$
$$= \sum_{\biguplus_{h=1}^{NO} S_h = U, |U| = NP} \sum_{\biguplus_{i=1}^{NO} T_i = V, |V| = NL} ILL_l'(NO, NL, NE_r, NE_\theta).$$

$ILL_l(NO, NP, NL, NE_r, NE_\theta) > 0$ when the statement $(|T_i \cap R_j| > \lceil |T_i|/2 \rceil) \wedge (|T_i \cap Q_k| > \lceil |T_i|/2 \rceil)$, which means at least two line segments of object i are superimposing on each other, is true. The nonempty endpoint multisets S_h and line segment multisets T_i are from Lemma III. The collection of nonempty multisets $m_j \times R_j$={line segments in r_l separated by '='s}, $1 \leq j \leq J$ with $m_j > 0$ and the collection of nonempty multisets $o_k \times Q_k$={line segments in θ_l separated by '='s}, $1 \leq k \leq K$ with $o_k > 0$ are two partitions of $\biguplus_{i=1}^{NO} T_i$.

Theorem III. For the r_l and θ_l in a line string $(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)$ with given numbers of objects $NO>0$, endpoints $NP \geq NO$, line segments $0 < NL \leq \binom{NP-NO+1}{2}$, equal signs NE_r between the lines in r_l with $0 \leq NE_r \leq NL-1$, and equal signs NE_θ between the lines

in θ_l with $0 \leq NE_\theta \leq NL-1$, the number of spatial relations NSR_l of (r_l,θ_l) is given by

$$NSR_l(NO,NP,NL,NE_r,NE_\theta)$$
$$= \sum_{\biguplus_{h=1}^{NO} S_h=U,|U|=NP} \quad \sum_{\biguplus_{i=1}^{NO} T_i=V,|V|=NL} NSR_l'''(NO,NL,NE_r,NE_\theta),$$
$$NSR_l'''(NO,NL,NE_r,NE_\theta)$$
$$= NSR_l''(NO,NL,NE_r)NSR_l''(NO,NL,NE_\theta)-$$
$$ILL_l'(NO,NL,NE_r,NE_\theta), \; and$$
$$ILL_l'(NO,NL,NE_r,NE_\theta)$$
$$= \begin{cases} 0 & if\ P\ is\ false \\ \left[\sum_{\biguplus_{j=1}^{J} R_j^{m_j} = \biguplus_{i=1}^{NO} T_i} \frac{(NL-NE_r)!}{\prod_{j=1}^{J} m_j!} \right] \left[\sum_{\biguplus_{k=1}^{K} Q_k^{o_k} = \biguplus_{i=1}^{NO} T_i} \frac{(NL-NE_\theta)!}{\prod_{k=1}^{O} o_k!} \right] & otherwise \end{cases}$$

where statement $P = (|T_i \cap R_j| > \lceil |T_i|/2 \rceil) \wedge (|T_i \cap Q_k| > \lceil |T_i|/2 \rceil)$, $\sum_{j=1}^{J} m_j = NL - NE_r$, and $\sum_{k=1}^{K} o_k = NL - NE_\theta$.

Lemma II and Theorem II are to find the number of spatial relations of (r_p,θ_p), so are Lemma III and Theorem III to (r_l,θ_l)'s. To find the number of spatial relations NSR_{pl} of $(r_p,\theta_p,r_l,\theta_l)$, we only need to integrate these lemmas and theorems into Theorem IV.

Theorem IV. For the r_p, θ_p, r_l and θ_l in a line string $(r_c,\theta_c,r_p,\theta_p,r_l,\theta_l)$ with given numbers of objects $NO>0$, endpoints $NP \geq NO$, line segments $0<NL \leq \binom{NP-NO+1}{2}$, equal signs NE_{r_p} between the endpoints in r_p with $0 \leq NE_{r_p} \leq NP-1$, equal signs NE_{θ_p} between the endpoints in θ_p with $0 \leq NE_{\theta_p} \leq NP-1$, equal signs NE_{r_l} between the lines in r_l with $0 \leq NE_{r_l} \leq NL-1$, and equal signs NE_{θ_l} between the lines in θ_l with

$0 \leq NE_{\theta_l} \leq NL\text{-}1$, the number of spatial relations NSR_{pl} of $(r_p,\theta_p,r_l,\theta_l)$ is given by

$$NSR_{pl}(NO, NP, NL, NE_{r_p}, NE_{\theta_p}, NE_{r_l}, NE_{\theta_l})$$
$$= \sum_{\biguplus_{h=1}^{NO} S_h = U, |U| = NP} NSR_p'''(NO, NP, NE_{r_p}, NE_{\theta_p})$$
$$\sum_{\biguplus_{i=1}^{NO} T_i = V, |V| = NL} NSR_l'''(NO, NL, NE_{r_l}, NE_{\theta_l}).$$

Collecting the above theorems, we reach the final theorem which gives the minimum number of spatial relations of a line string.

Theorem V. For a line string $(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)$ with given numbers of centroids $NC \geq 0$, objects $NO \geq 0$, endpoints $NP \geq NO$, line segments $0 \leq NL \leq \binom{NP-NO+1}{2}$, and equal signs $0 \leq NE \leq 2NC+2NP+2NL\text{-}6$, where

$$\begin{cases} NO > 0 & if\ NC = 0 \\ NP = 0 \wedge NC > 0 & if\ NO = 0, \\ NC = NO & otherwise \end{cases}$$

the number of spatial relations NSR_{ls} of the line string $(r_c, \theta_c, r_p, \theta_p, r_l, \theta_l)$ is given by

$$NSR_{ls}(NC, NO, NP, NL, NE)$$

$$\in \begin{cases} [\sum NSR_c(NC, NE_{r_c}, NE_{\theta_c})] & if\ NP = 0 \\ [\sum NSR_p(NO, NP, NE_{r_p}, NE_{\theta_p}), \\ \quad \sum NSR_c(NC, NE_{r_c}, NE_{\theta_c}) \\ \quad\quad \sum NSR_p(NO, NP, NE_{r_p}, NE_{\theta_p})] & if\ NL = 0 \\ [\sum NSR_{pl}(NO, NP, NL, NE_{r_p}, NE_{\theta_p}, NE_{r_l}, NE_{\theta_l}), \\ \quad \sum NSR_c(NC, NE_{r_c}, NE_{\theta_c}) \\ \quad\quad \sum NSR_{pl}(NO, NP, NL, NE_{r_p}, NE_{\theta_p}, NE_{r_l}, NE_{\theta_l})] & otherwise \end{cases}$$

where the sum, denoted by Σ, is taken over all nonnegative integers $NE_{r_c} \leq NC - 1$, $NE_{\theta_c} \leq NC - 1$, $NE_{r_p} \leq NP - 1$, $NE_{\theta_p} \leq NP - 1$, $NE_{r_l} \leq NL - 1$, $NE_{\theta_l} \leq NL - 1$, and $\sum_{w \in \{r_c, \theta_c, r_p, \theta_p, r_l, \theta_l\}} NE_w = NE$.

Figure 13 shows $NSR_{ls}(0,2,3,1,0) \geq \sum_{S_h} \left(\frac{3!}{1!2!}\right)\left(\frac{3!}{1!2!}\right)\left[\sum_{T_i}\left(\frac{1!}{1!}\right)\left(\frac{1!}{1!}\right)\right] = 18$ spatial relations of a line string with *NC=0*, *NO=2*, *NP=3*, *NL=1*, and *NE=0*.

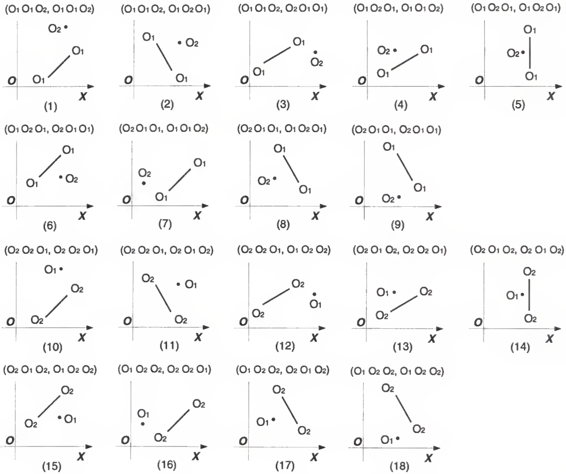


Figure 13. The minimum eighteen spatial relations of a line string with *NC=0*, *NO=2*, *NP=3*, *NL=1*, and *NE=0*.

5.2 Space Needs

The space needs of an image's line string depend on the image's complexity, the threshold value of algorithm *APPROX_LINE*, and the objects' configurations. After the linear image is generated, the space needs of its line string can be found as follows. Assume each symbol or number needs one byte of memory. The maximum space needs for a line string with *NC* centroids, *NP* endpoints, *NL* line segments, and *NE* equal signs

are $2(NC+NP+NL)+NE+3$ bytes. Without considering the equal sign '=', a line string needs NC bytes for length r_c, the NC bytes for angle θ_c, and one byte for the delimiter between r_c and θ_c for NC centroids. Therefore, $2NC+1$ bytes are for NC centroids in a line string when $NC \geq 0$. Under the same reasoning, NP endpoints need $2NP+1$ bytes and NL lines require $2NL+1$ bytes. Hence, the space needs are $2(NC+NP+NL)+NE+3$ bytes since NE equal signs need NE bytes. Figure 14 displays the relationship between space needs and numbers of spatial relations NSR of line strings. The NSR increases abruptly because it is in $O[(space/n)!]$ where n is a positive integer. These numbers, created by projection, are conservative. The actual numbers should be much higher than these projected values. This graph shows the maximum space of 16 bytes and lacks the genuine relationship among NSR_{ls}, NSR_c, NSR_p, and NSR_{pl}. We would expect NSR_{ls} to have the largest value when space exceeds a threshold value, which would not be large. Second comes NSR_{pl}, third NSR_c, and last NSR_p.

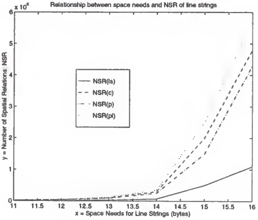


Figure 14. Relationship between space needs and numbers of spatial relations *NSR* of line strings.

5.3 Comparative Performance Analysis

This section augments the empirical analyses of Chang and Lee [8] in calculating

the storage space and matching speed of line strings. Assume the image object has a rectangular shape. Since a rectangular object in a line string needs two endpoints or one if it is square, the maximum number of symbols for N rectangular objects is $8N+3$: [N(centroids)+$2\times N$(two endpoints for an object)+N(one line segment for an object)]$\times 2$(one for length r and one for angle θ)+3(delimiters). Table 1 shows the number of symbols required among various string image data structures. The storage space of $8N+3$ bytes is more than RCOS string's $5N$ bytes for N object symbols. Line strings, however, provide more spatial information than RCOS strings.

Table 1. Storage space and its ratios of various string image data structures. B: 2D B-string, C: 2D C-string, G: 2D G-string, R: RCOS string, and L: Line string.

Number of objects (N)	**Number of symbols**					**L/B**	**L/C**	**L/G**	**L/R**
	2D B-string	**2D C-string**	**2D G-string**	**RCOS string**	**Line string**				
2	8.0	7.6	10.0	10.0	19.0	2.38	2.50	1.90	1.90
4	16.0	19.6	43.6	20.0	35.0	2.19	1.79	0.80	1.75
8	32.0	57.2	173.2	40.0	67.0	2.09	1.17	0.39	1.68
16	65.6	164.4	672.0	80.0	131.0	2.00	0.80	0.19	1.64
32	141.0	476.8	2673.2	160.0	259.0	1.84	0.54	0.10	1.62
64	291.4	1422.0	9626.0	320.0	515.0	1.77	0.36	0.05	1.61
128	635.6	4768.8	32479.	640.0	1027.0	1.62	0.22	0.03	1.60

The dispositions of two rectangular objects can create 169 types of spatial relationships [8]. Table 2 contains the number of operations required to retrieve any of the 169

relations for three string image data structures.

Table 2. Computational speed in retrieving one of the *169* types of spatial relations.

Number of operations	**2D B-string**	**RCOS string**	**line string**
No. of comparisons	0	6	16
No. of subtractions	16	0	0
No. of multiplications	8	0	0

Compared to other string image data structures, the drawback of line strings may arise from the increased need for storage space or a slow down in matching speed. Depending on a user's need, the above statistics demonstrate that the effective spatial representation property of line string may compensate for the increase in space or computational speed.

CHAPTER 6
A LONGEST APPROXIMATE COMMON SUBSEQUENCE PROBLEM

Finding a longest common subsequence of two strings occurs in a number of computing and data-processing applications. A classical *longest common subsequence* problem [21] (abbreviated LCS) is, given two strings X and Y, to find a maximum length common subsequence of X and Y. A subsequence of a given string is just the given string with some symbols (possibly none) left out. String Z is a common subsequence of X and Y if Z is a subsequence of both X and Y. Finding an LCS is mainly used to measure the discrepancies between two strings. An LCS, however, does not always reveal the degree of difference between two strings that some problems require. For example, if $s_0 = \langle a, b \rangle$, $s_1 = \langle b, b \rangle$ and $s_2 = \langle b, a \rangle$, an LCS $\langle b \rangle$ of s_0 and s_1 is the same as an LCS of s_0 and s_2. From the viewpoint of LCS, the resemblance of s_1 and s_0 is the same as the resemblance of s_2 and s_0. However, s_2 has more symbols in common with s_0 than s_1 does, although not in the same order. Approximating an LCS may better characterize the discrepancies between two strings.

6.1 Definition

A *longest approximate common subsequence* problem (abbreviated LACS) produces a maximum-gain approximate common subsequence of two strings. An approximate subsequence of a string X is a string edited from a subsequence of X. The only editing operation allowed here is an adjacent symbol interchange. String Z is an approximate common subsequence of two strings X and Y if Z is an approximate subsequence of both X and Y. The gain function g, which will be described later, assigns a nonnegative

real number to each subsequence. Formally, the LACS problem is defined as follows: Given two strings X and Y, a weight $W_m>0$ for a symbol in an approximate common subsequence, and a weight $W_s \leq 0$ for an adjacent symbol interchange operation, a string Z is a longest approximate common subsequence of X and Y if Z satisfies the following two conditions:

1) Z is an approximate common subsequence of X and Y, and
2) the gain $g(X,Y,Z,W_m,W_s) = |Z|W_m+\delta(X,Z)W_s+\delta(Y,Z)W_s$ is maximum among all approximate common subsequences of X and Y, where $\delta(X,Z)$ is the minimum edit distance from a subsequence of X to Z, so is $\delta(Y,Z)$ to Y and Z.

A string Z is said to be of edit distance k to a string Z' if Z can be transformed to be equal to Z' with a minimum sequence of k adjacent symbol interchanges. The following is an LACS example. Let $X = \langle B, A, C, E, A, B\rangle$, $Y = \langle A, C, D, B, B, A\rangle$, $W_m = 3$, and $W_s = -1$. A longest approximate common subsequence of X and Y is $Z = \langle A, B, C, B, A\rangle$ with the gain $g(X,Y,Z,W_m,W_s) = |Z|W_m+\delta(X,Z)W_s+\delta(Y,Z)W_s = 5\times 3+2\times(-1)+1\times(-1) = 12$.

6.2 Computational Complexity

This section breaks up the complication of the LACS by sorting the problem into LACS_i, $0 \leq i \leq \min\{|X|,|Y|\}-1$ categories, where X and Y are input strings. Subsequently, we prove that $\text{LACS}_{\min\{|X|,|Y|\}-1}$ problem is NP-hard and conjecture that the problems from LACS_1 to $\text{LACS}_{\min\{|X|,|Y|\}-2}$ are at least as hard as the NP-complete problems.

6.2.1 LACS Categories

The LACS problem fits into categories according to the relation between the weights $W_m > 0$ and $W_s \leq 0$:

- $LACS_0$—when $0 < W_m \leq -W_s$, $LACS_0$ is reduced to an LCS problem since no adjacent symbol interchanges are allowed for any symbol in $LACS_0$;
- $LACS_1$—when $-W_s < W_m \leq -2W_s$, any symbol in $LACS_1$ makes no more than 1 adjacent symbol interchange;
- $LACS_2$—when $-2W_s < W_m \leq -3W_s$, any symbol in $LACS_2$ makes no more than 2 adjacent symbol interchanges;
- $LACS_i$—when $-iW_s < W_m \leq -(i+1)W_s$, any symbol in $LACS_i$ makes no more than i adjacent symbol interchanges; and
- $LACS_{\min\{|X|,|Y|\}-1}$—when $-(\min\{|X|,|Y|\}-1)W_s < W_m$, X and Y are input strings, any symbol in $LACS_{\min\{|X|,|Y|\}-1}$ makes no more than $\min\{|X|,|Y|\}-1$ adjacent symbol interchanges, which is the maximum number of interchanges a symbol is allowed to make.

Another useful abbreviation is that an $LACS_i(X,Y)$ equals an $LACS_i$ of X and Y. The $LACS_i$ problem can be interpreted in another way called a *trace* [49]. Diagrammatically aligning the input strings X and Y and drawing lines from symbols in X to their matches in Y provides the trace of X and Y. Figure 15 illustrates the example in Section 6.1 through trace. In an $LACS_i$ trace, each line is allowed to have a maximum of i line-crossings, i.e. the symbol touched by the line may make no more than i adjacent symbol interchanges. The total number of line-crossings in a trace is $\delta(X,Z) + \delta(Y,Z)$.

$LACS_2(X,Y) = Z = \langle A\ B\ C\ B\ A \rangle$

$g(X,Y,Z,3,-1) = 5 \times 3 + 3 \times (-1) = 12$

Figure 15. An $LACS_2$ illustrated through trace

6.2.2 The $LACS_{\min\{|X|,|Y|\}-1}$ Problem is NP-hard

In Theorem VI, we show that any instance of extended string-to-string correction problem, which was proven to be an NP-complete problem by Wagner in 1975 [48], can be reduced in polynomial time to an instance of $LACS_{\min\{|X|,|Y|\}-1}$. The extended string-to-string correction problem (ESSCP)—given finite alphabet Σ, two strings X and $Y \in \Sigma^*$, and a positive integer k—determines whether there is a way to derive the string Y from the string X by a sequence of k or fewer operations of single symbol deletion or adjacent symbol interchange.

Theorem VI ($LACS_{\min\{|X|,|Y|\}-1}$ is NP-hard)

If X and Y are input strings, then the $LACS_{\min\{|X|,|Y|\}-1}$ problem is NP-hard.

Proof We first show that $LACS_{\min\{|X|,|Y|\}-1}$ does not belong to NP. Given an instance of the problem, we use as a certificate an $LACS_{\min\{|X|,|Y|\}-1}$ Z of X and Y. A verification algorithm checks if the gain $g(X,Y,Z,W_m,W_s) = |Z|W_m+\delta(X,Z)W_s+\delta(Y,Z)W_s \geq k'$, a nonnegative real number. From the ESSCP, we know it is unlikely to find $\delta(X',Z)$ and $\delta(Y',Z)$ in polynomial time, where X' and Y' are subsequences of X and Y, respectively. If $\delta(X',Z)$ and $\delta(Y',Z)$ cannot be found in polynomial time, then $\delta(X,Z)$ and $\delta(Y,Z)$ definitely cannot be found in polynomial time. Therefore, we could say with certainty that a polynomial-time verification algorithm does not exist.

To prove that $LACS_{\min\{|X|,|Y|\}-1}$ is NP-hard, we show that ESSCP $\leq_p$ $LACS_{\min\{|X|,|Y|\}-1}$. In other words, any instance of ESSCP can be reduced in polynomial time to an instance of $LACS_{\min\{|X|,|Y|\}-1}$. Let the two input strings be the same at both problems. We now show that string X needs k operations of deletion or interchange to derive string Y if and only if the corresponding $LACS_{\min\{|X|,|Y|\}-1}(X,Y)$ problem has an LACS Z with a gain $g(X,Y,Z,W_m,W_s) \geq |Y|W_m+(k-|X|+|Y|)W_s$. Suppose that string X needs k

operations of single symbol deletion or adjacent symbol interchange to derive string Y. The number of deletions has to be $|X|-|Y|$, making the number of interchanges $k-|X|+|Y|$. Thus, for an $\mathrm{LACS}_{\min\{|X|,|Y|\}-1}(X,Y)$ trace, there are $|Y|=|Z|$ lines (matchings) and $k-|X|+|Y|$ line-crossings (interchanges). Therefore, the gain $g(X,Y,Z,W_m,W_s)$ is $|Y|W_m+(k-|X|+|Y|)W_s$. Conversely, suppose that $Z = \mathrm{LACS}_{\min\{|X|,|Y|\}-1}(X,Y)$ has a gain $g(X,Y,Z,W_m,W_s) = |Y|W_m+(k-|X|+|Y|)W_s$. Because X has $|Y|$ symbols in common with Y, Z could equal Y. X then needs $|X|-|Y|$ deletions and $k-|X|+|Y|$ interchanges to derive Z, i.e., Y. Thus, for the ESSCP, string X needs $|X|-|Y|$ deletions and $k-|Y|+|X|$ interchanges to derive string Y. The total number of operations required to derive Y from X, therefore, is $k = (|X|-|Y|)+(k-|Y|+|X|)$. □

CHAPTER 7
A HEURISTIC APPROXIMATION ALGORITHM

Despite the unlikelihood of finding a polynomial-time algorithm for solving the LACS problem exactly, near-optimal solutions in polynomial time may still be possible.

7.1 ALACS Algorithm

An approximation algorithm finds an approximate LACS (abbreviated ALACS) of two strings. The procedure APPROX_LACS repeatedly calls an LCS-routine $\lceil W_m/\text{-}W_s \rceil$ times, beginning with allowing zero number of edit operations for each selected symbol. Each round, the allowable number of edit operations is increased by one. The symbols selected are marked off from the input strings to prevent consideration in the next round. Suppose X and Y are input strings, and the trace T is empty and $i = 0$ at first. It executes the following steps.

1. Find an LCS of X and Y.
2. Select symbols from the LCS such that each symbol makes no more than i line-crossings in trace T.
3. Eliminate the selected symbols from X and Y, and add them to T.
4. $i = i + 1$.
5. Repeat the above steps until $i \geq \lceil W_m/\text{-}W_s \rceil$.

The procedure uses two arrays $m[0..|X|]$ and $n[0..|Y|]$ to flag which symbols in X and Y are selected. If a symbol is selected, it stores the index of matching symbol of the other string. If the symbol is not selected, it stores 0. An LCS function *LCS_LENGTH* is

borrowed from Cormen et al. [14], and is modified to include checking whether symbols in X and Y are selected.

```
APPROX_LACS (X, Y, Wm, Ws)
        /* X and Y: input strings */
        /* Wm: weight for a symbol in an approximate common subsequence */
        /* Ws: weight for an adjacent common symbol interchange operation */
    for h ← 0 to ⌈Wm/-Ws⌉ - 1
        do for i ← 1 to |X|
                do m[i] ← 0
            for j ← 0 to |Y|
                do n[j] ← 0
            LCS_LENGTH (X, Y, m, n, b)
            SELECT_SYM (X, Y, |X|, |Y|, m, n, b, h)
```

Procedure SELECT_SYM finds an LCS and selects symbols from it. It makes sure the selected symbols do not make more than the maximum allowable number of edit operations.

```
SELECT_SYM (X, Y, i, j, m, n, b, h)
    if i = 0 or j = 0
        then return
    if b[i,j] = ↖
        then if ¬OVER_CROSSING(X, Y, i, j, m, n, h)
                then m[i] ← j
                     n[j] ← i
            SELECT_SYM(X, Y, i-1, j-1, m, n, b, h)
```

```
    elseif b[i,j] = '↑'
        then SELECT_SYM(X, Y, i-1, j, m, n, b, h)
    else SELECT_SYM(X, Y, i, j-1, m, n, b, h)
```

Procedure OVER_CROSSING checks a line from the *i*th symbol of *X* to the *j*th symbol of *Y* does not cross more than *h* other lines in trace.

```
OVER_CROSSING (X, Y, i, j, m, n, h)
    h1 = 0
    for i1 ← 1 to i-1
        do if m[i1] > j
            then h1 = h1 + 1
    for j1 ← 1 to j-1
        do if n[j1] > i
            then h1 = h1 + 1
    if h1 > h
        then return TRUE
        else return FALSE
```

For this approximation algorithm, the running time is $O(\lceil W_m/-W_s \rceil (|X|+|Y|)^2)$ and the space needs is $O(|X||Y|)$ for the *LCS_LENGTH* function. Figure 16 illustrates the progress of the approximation algorithm on an instance. We name the trace *T* after

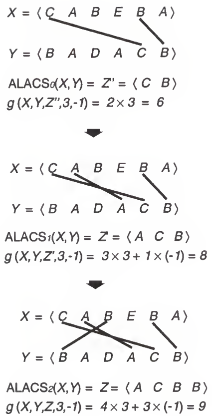


Figure 16. A sequence of $ALACS_{0..2}$ produced by APPROX_LACS on an instance.

executing the *i*th LCS-routine as an $ALACS_{i-1}$, $1 \le i \le \min\{|X|,|Y|\}$. The $ALACS_i$ means a symbol in $ALACS_i$ can make no more than *i* adjacent symbol interchanges. Figure 17 is an instance of an $LACS_2$ problem. The difference between the optimal gain and approximate gain is *1*.

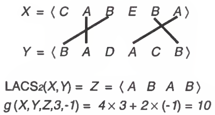


Figure 17. An $LACS_2$

7.2 Ratio Bound

For a maximization problem, we say that an approximation algorithm for the problem has a *ratio bound* $\rho(n)$ if for any input of size n, the gain G of the solution produced by the approximation algorithm is within a factor of $\rho(n)$ of the gain G^* of an optimal solution, namely, $G^*/G \leq \rho(n)$. For an $ALACS_i(X,Y)$, $0 \leq i \leq \min\{|X|,|Y|\}-1$, problem, the worst ratio bound is:

$$\rho_i(n) = \frac{max\ gain\ of\ LACS_i}{min\ gain\ of\ ALACS_i}. \tag{30}$$

The next two lemmas show how to compute the minimum gain of $ALACS_i$ and the maximum gain of $LACS_i$, respectively.

Lemma IV (Minimum gain of $ALACS_i$)

If Z is an LCS of strings X and Y, the minimum gain of an $ALACS_i(X,Y)$, $0 \leq i \leq \min\{|X|,|Y|\}-1$, is $|Z|W_m$ for any i.

Proof The procedure APPROX_LACS is implemented by calling the LCS-routine $\lceil W_m/\text{-}W_s \rceil$ times. Each time, they select symbols from the LCS, add them to the ALACS, and remove them from input strings. Since it selects every symbols from the first LCS, the minimum gain of the approximation algorithm is $|Z|W_m$. □

Before discussing Lemma V, some terminologies need introduction. A set of lines is completely-crossing if each line crosses every other line in the set. If a set of lines is independent, then every line in the set does not cross any line in the other set.

Lemma V (Maximum gain of $LACS_i$)

If Z is an LCS of strings X and Y, the maximum gain of an $LACS_i(X,Y)$, $0 \leq i \leq \min\{|X|,|Y|\}-1$, is $(i+2)|Z|W_m/2$.

Proof We show that an $LACS_i$ has a maximum gain when there are $|Z|$ independent,

completely-crossing and $i+1$-line sets in its trace. The requirement can be broken into four conditions which are examined separately.

1. $|Z|$ sets: Every independent set contributes at least one symbol to an LCS. If the number of independent sets is more than $|Z|$, then the length of the LCS of X and Y is longer than $|Z|$. It contradicts the assumption that Z is an LCS of X and Y. On the other hand, the gain is not maximum if the number of independent sets is less than $|Z|$ since each set contributes a fixed gain, which will be explained later. Therefore, the number of independent sets is $|Z|$ when $LACS_i$ has a maximum gain.
2. Completely-crossing: Suppose one independent set is not completely-crossing, then there must have some lines not crossing one another, i.e., the lines are parallel in the sense of LCS. Each of the parallel lines contributes one symbol to an LCS. Since the length of the LCS is fixed, an $LACS_i$ achieves the maximum total gain only by having the maximum gain from each line. For an $LACS_i$, the maximum gain one line can reach is when the set of this line and other i lines is completely-crossing. Therefore, each set has to be completely-crossing in order to have the maximum gain of an $LACS_i$.
3. Independent: Every set is independent, otherwise it is not completely-crossing.
4. $i+1$-line: From the above discussion, we know each set is completely-crossing. For a line in the set, it is only allowed to have no more than i line-crossings because of the limitation of $LACS_i$. Therefore, the maximum lines (also the maximum gain) a set can own is $i+1$.

In short, there are $|Z|$ sets in the trace, where each set has $i+1$ lines and $1+2+ \ldots +i = i(i+1)/2$ line-crossings, when an $LACS_i$ has a maximum gain. Consequently, the max-

imum gain of an $LACS_i$ is $|Z|[(i+1)W_m+i(i+1)W_s/2] \leq (i+2)|Z|W_m/2$ since $-iW_s<W_m\leq-(i+1)W_s$. □

Theorem VII gives the worst ratio bound of $ALACS_i$ by applying the above two lemmas. Since we take a very conservative approach to find $\rho_i(n)$, the actual ratio bound is expected to be much less than $(i+2)/2$ when $i > i_0$, a positive constant.

Theorem VII (The worst ratio bound of $ALACS_i$)

The worst ratio bound $\rho_i(n)$ of an $ALACS_i(X,Y)$, $0\leq i\leq\min\{|X|,|Y|\}-1$, is $(i+2)/2$ for any input size n.

Proof From Equation 30 and Lemmas I and II, the worst ratio bound $\rho_i(n)$ of an $ALACS_i(X,Y)$, $0\leq i\leq\min\{|X|,|Y|\}-1$, is $[(i+2)|Z|W_m/2]/(|Z|W_m) = (i+2)/2$ no matter what the input size n is. □

CHAPTER 8
AN OPTIMIZATION NEURAL NETWORK

A modified Hopfield neural network is designed to solve the LACS problem. A technique of interception is used to determine the values of network coefficients.

8.1 The Hopfield Network

Hopfield [22] discovered a Liapunov function as the energy function of the network:

$$E = \left(-\frac{1}{2}\right) \sum_{i} \sum_{j} W_{ij} O_i O_j - \sum_{j} I_j O_j + \sum_{j} \theta_j O_j$$

In solving an LACS problem, this energy function is compared with another function built from LACS constraints in order to determine the network weights. Let strings $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$ be an instance of an LACS problem. The Hopfield net involves mn units represented as an $m \times n$ array. The energy function constructed from the LACS problem constraints is

$$E = E_1 + E_2 + E_3$$

where

$$E_1 = \frac{A}{2} \sum_{x} \left(\sum_{y} g_{xy} O_{xy} - h1_x \right)^2,$$
$$E_2 = \frac{B}{2} \sum_{y} \left(\sum_{x} g_{xy} O_{xy} - h2_y \right)^2, \text{ and}$$
$$E_3 = \frac{C}{2} \sum_{x} \sum_{y} \sum_{x'=x+1}^{|X|} \sum_{y'=1}^{y-1} O_{xy} O_{x'y'}.$$

A, B and C are constants. O_{ij}, with $1 \leq i \leq m$ and $1 \leq j \leq n$, indicates whether x_i matches y_j. Functions g_{ij}, $h1_i$ and $h2_j$ are

$$g_{ij} = \begin{cases} 0 \text{ if } x_i \neq y_j \\ 1 \text{ if } x_i = y_j \end{cases},$$

$$h1_i = |Y|_{x_i} / |X|_{x_i}, \text{ and } h2_j = |X|_{y_j} / |Y|_{y_j},$$

where $|X|_{x_i}$ is the number of symbol x_i in string X. E_1 and E_2 reflect the constraints that each row (X) or column (Y) contains a fraction $h1$ or $h2$ of a single 1. E_3 reflects the constraint that the minimum number of line-crossings is favored. By comparing this energy function with the Liapunov function, the weight and the external input are given by

$$W_{xy,x'y'} = -Ag_{xy}g_{xy'}\delta_{xx'} - Bg_{xy}g_{x'y}\delta_{yy'} -$$

$$C\lambda_{xx'}\lambda_{y'y} \text{ and}$$

$$I_{xy} = Ag_{xy}h1_x + Bg_{xy}h2_y + \theta_{xy}$$

where

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \text{ and } \lambda_{ij} = \begin{cases} 0 & \text{if } i \geq j \\ 1 & \text{if } i < j \end{cases}.$$

Some results are not valid when the input string is beyond a certain length, e.g. about 7. The validity can be improved by changing the coefficient values, but this may have the undesirable effect of sacrificing the gain. Therefore, the validity of results is checked after convergence, and two actions are taken to preserve the validity. When a line makes more than $\lceil W_m/-W_s \rceil - 1$ line-crossings, it is canceled; and when a symbol is picked more than once, only the first pick is accounted for.

8.2 Coefficient Values Determination by Interception

It has been observed that the convergence and the results of Hopfield net to the LACS problem is highly dependent upon the coefficients, and different input strings may have different optimal coefficient values. The values of coefficients A and B relative to the value of C affect the net, i.e. A/C (or B/C) affects the net. With sufficiently large values for A and B, the low-energy states will represent valid results and the maximum number of matchings, while a large value for C ensures a minimum number of line-crossings. The threshold value θ_j is fixed. Thus only the value of coefficient A needs to be decided. Figure 18 shows a typical curve of gains and coefficient values. It resembles the shape of

a trapezoid with the top slowing declining, and eventually becoming a constant. The value of coefficient A is determined by the interception of two lines, which are extrapolated from the two sides of the curve. From some random instance experiment, the peak of the curve occurs at about $A = 10$, and the curve becomes constant at $A \geq 200$. So $A = 1$ and $A = 5$ are picked for deciding the line on the left hand side of the curve, and $A = 100$ and $A = 200$ are picked for the other line. It turns out that the value of A is almost always selected correctly. This is because the curve shown in Figure 18 applies to most strings.

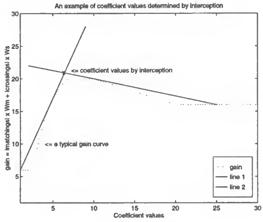


Figure 18. An example of a coefficient value determined by interception.

CHAPTER 9
LACS ANALYSIS AND APPLICATIONS

A comparative performance of the two LACS approximation methods: the heuristic algorithm and the Hopfield neural network is given in this Chapter. This Chapter also lists some possible applications of the LACS method.

9.1 Comparative Performance of Approximation Methods

Table 3 Gains and running time of string matching algorithms with $W_m=3$ and $W_s=-1$ on some random instances.

|LACS|	LCS	ALACS		Hopfield NN		LACS
	|LCS|×3	gain	time	gain	time	gain
1	3	3	0.16	3	0.23	3
2	3	5	0.16	5	0.23	5
3	6	8	0.17	8	0.22	8
4	9	11	0.15	11	0.21	11
5	9	10	0.17	12	0.30	13
6	12	15	0.16	14	0.36	15
7	12	16	0.17	11	0.33	16
8	18	20	0.18	20	0.64	20
9	12	13	0.18	13	0.53	22
10	24	25	0.16	25	0.89	25

Table 3 lists the gains and running time of LCS, ALACS, Hopfield net, and LACS methods with $W_m=3$ and $W_s=-1$ on some random instances. Figure 19 draws a relation of gains and running time of LCS, ALACS, Hopfield net, and LACS methods with $W_m=6$ and $W_s=-1$ on some random instances. The output gains of ALACS and Hopfield net

are pretty much the same, though the Hopfield net is a little bit poorer. The Hopfield net is much slower, but it requires less memory. The relation distance between ALACS (or Hopfield net), and also the distance between ALACS (or Hopfield net) and LCS, increases with increasing $W_m/-W_s$ or |LACS|. The experimental results also show the ratio bound $\rho_i(n) = (i+2)/2$ of an $ALACS_i$ is exaggerated.

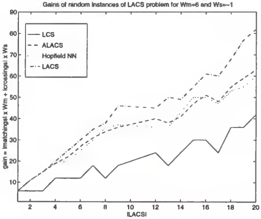


Figure 19. Gains and running time of string matching algorithms with W_m=*6* and W_s=*-1* on some random instances.

9.2 Possible LACS Applications

Kruskal [32] suggested some possible application categories for sequence comparison, which LACS belongs to.

1. Molecular biology: The main use of sequence comparison in molecular biology is to detect and characterize the homology (i.e., correspondence) between two or more related sequences. [50] Two structures are said to be homologous if they have a common evolutionary ancestor, and are said to have a high degree of homology if the differences that have developed between them are relatively minor.
2. Human speech: Consider the much-studied problem of recognizing an isolated word selected from a limited vocabulary. A very common approach is to provide the

recognizer with one or several versions of each vocabulary word in sequence form. To recognize an unknown word, the recognizer converts it to sequence form, compares it to many stored sequences, and selects the one that matches best.[15]

3. Computer science: Sequence comparison has been used in many computer science applications. [20] String-editing and file-comparison are two among the many applications. String-editing is the correction of human errors made at the input stage. The file-comparison, where each line or record of the file may be treated as a single symbol, is now in routine use.
4. Codes and error control: Strings of symbols are used extensively to code information for transfer over a noisy channel such as radio, telegraph, microwave transmission, and so forth. Since noise in the channel introduces error into the received signal, many questions arise about detection and correction of error. Though the primary focus of coding theory has always been on substitution errors (in which the wrong symbols are received), it is also true that insertion and deletion errors (in which too many or too few symbols are received) have long been studied under the heading of synchronization error. Levenshtein [37] presented a distance function that is appropriate in the presence of insertion and deletion errors. His distance function plays a major role in sequence comparison.
5. Bird song: Bird song is a complex, learned behavioral pattern. The development of bird song shows parallels with language acquisition in humans. Sequence comparison analysis of bird songs produces a useful measure of distance between pairs of songs. [6]
6. Human depth perception: The human visual system fuses images from the two eyes during binocular depth perception is a form of sequence comparison. [27]
7. Some other discrete-time applications:

a. Geological strata: In geology, stratigraphic sequences can be obtained from drill holes and outcroppings. Each stratum may simply be classified as to type, or the classification may be supplemented by additional information about thickness, geochemical or mineral assay, fossils, etc. "Correlation" of different sequences is frequently necessary; i.e., it is necessary to determine which strata in one such sequence correspond geologically to which strata in another. [45]

b. Tree rings: Dendrochronology refers to the science of dating based on tree rings. [16] It requires the comparison of sequences obtained from different logs, most notably from logs of the bristle-cone pine.

c. Varves: Varves are annual layers of sediment, generally clay, in which it is possible to count the years, typically due to variation between the summer and winter sediment.[23] Varve chronology requires comparison of sequences from different cores or different locations.

d. Text collation: Collation of different versions of the same text offers another possible application.

8. Some other continuous-time applications:

   a. Handwriting: Time-warping can be used in computer processing handwritten material such as signatures and line drawings.[46] Typically, the penpoint is tracked during the act of writing or drawing, with pressure of pen on paper or height of pen as a third coordinate, so that the record consists of the penpoint trajectory as a function of time.

   b. Evoked potential waves: Evoked potentials from the brain ("brain waves" in response to a stimulus) are known to vary somewhat in timing from one occasion to another, and might constitute an application of time-warping.

CHAPTER 10
A PROTOTYPE IMAGE RETRIEVAL SYSTEM

To realize the advantages of the above methods, a prototype traffic sign retrieval and browsing system is available on the World Wide Web, at http://www.cise.ufl.edu/~wenchen/idb. The experiment results show any image in the system can be precisely retrieved as long as each image has a different image string. The accuracy of image retrieval depends on the query details.

10.1 System Overview

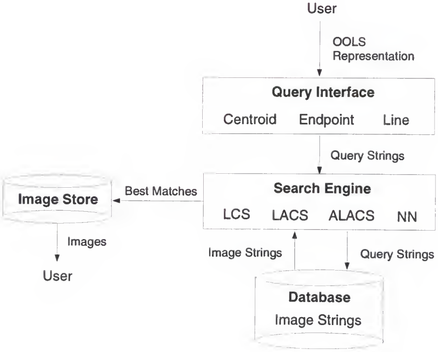


Figure 20. System structure.

Figure 20 presents the system structure. A query string need not match every image string in the database. An image string has to pass through a filter before it is qualified for matching. An image string passes through the filter only if it includes all image objects specified in the query string. The system interface is shown in Figure 21.

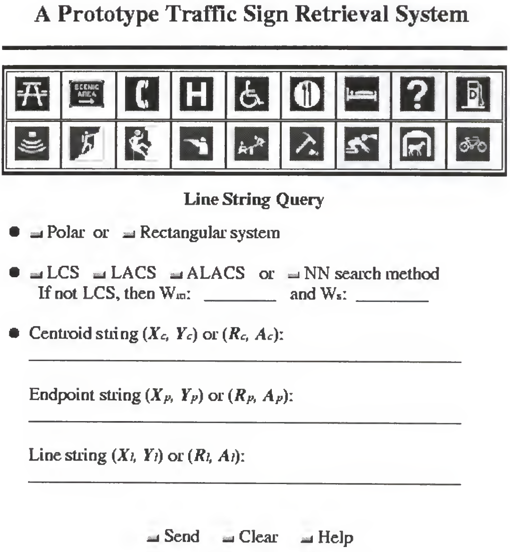


Figure 21. System interface.

The interface provides the following choices:

1. Polar or rectangular coordination system;

2. LCS, LACS, ALACS, or Hopfield neural network search method; and
3. Centroid, endpoint, or line string.

So far the system is not able to handle equal signs in strings since the search methods do not cover them. If a user chooses the LACS search method, a lengthy or indefinite waiting time might occur, because the LACS method is not a polynomial-time algorithm. The number of elements in an input string for the Hopfield neural network can not exceed a certain number, about *32*. The number of elements is restricted due to the memory limitation. The Hopfield net involves mn units represented as an $m \times n$ array, where m and n are the lengths of a query string and an image string, respectively.

10.2 Sample Images

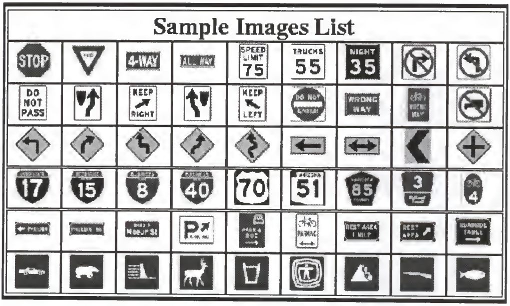


Figure 22. Some traffic sign samples.

Figure 22 lists some traffic signs stored in the system. The system processes *276* signs among the hundreds of possible traffic signs. The other traffic signs are either too complicated to be segmented or contain text only which need not be tested again.

Figure 22 shows a traffic sign and its skeletal, linear, and centroidal images. It also

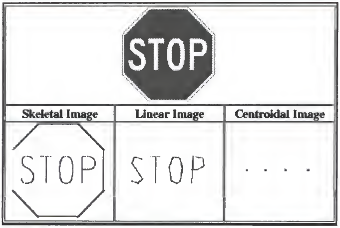


Figure 23. A stop sign and its related images.

Table 4 Image strings of Figure 23 stored in the database.

	
(X_c, Y_c)	(S T octagon O P, O = T = P = octagon = S)
(X_p, Y_p)	(S = S S S S S T T = T T O O O O O O O P = P P P P P, O = S T = P S = O O S P = P O O P S S P O O = S T = P = T = T)
(X_l, Y_l)	(S P S O S = O O O P T P O S O O S P P T, T = P O P = S O = S O P S S P O = O O O S T = P)
(r_c, Θ_c)	(S T octagon O P, P O octagon T S)
(r_p, Θ_p)	(S S S T S S S O O T O O T T O P O O P P P P P, P O O P P P P O O P T O O O T T S S T S S S S)
(r_l, Θ_l)	(O P S S O S O P P S P O T O O O S T P, T = P P O S O S O P T O P O S O S S P O)

illustrates the uncertain property of the curve-to-line transformation. Letter 'T' has a perfect transformation of two line segments. For the letters 'S', 'O' and 'P', users have no idea precisely how many lines or what kind of lines would be created from these letters.

The experiment shows the fuzzy computation doesn't hurt the system performance much, since the system also uses fuzzy algorithms to search for the best matches. The image strings of a stop sign stored in the database are shown in Table 4. The traffic signs themselves are stored in an image store. The endpoint and line strings do not include the octagon frame. The frame does not contribute many distinct features, but instead makes the strings look more complicated.

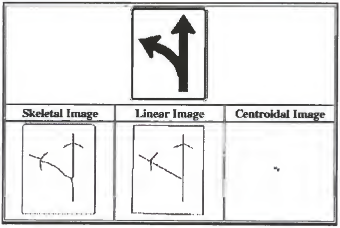


Figure 24. A traffic sign and its related images.

Table 5 Image strings of Figure 24 stored in the database.

	
(X_c, Y_c)	(arrow rectangle, rectangle arrow)
(X_p, Y_p)	(arrow arrow_head arrow_head arrow_head arrow_head arrow arrow_head = arrow = arrow arrow_head arrow_head, arrow arrow arrow_head arrow_head arrow arrow_head arrow_head arrow_head arrow_head arrow_head arrow)
(X_l, Y_l)	(arrow arrow_head arrow_head arrow arrow_head arrow_head, arrow_head arrow_head arrow_head arrow_head arrow arrow)

Table 5 (Continued) Image strings of Figure 24 stored in the database.

(r_c, Θ_c)	(arrow rectangle, rectangle arrow)
(r_p, Θ_p)	(arrow_head arrow_head arrow arrow arrow arrow_head arrow_head arrow_head arrow_head arrow_head arrow, arrow arrow arrow_head arrow_head arrow_head arrow_head arrow arrow_head arrow_head arrow_head arrow)
(r_l, Θ_l)	(arrow_head arrow_head arrow_head arrow_head arrow arrow, arrow_head arrow_head arrow arrow_head arrow arrow_head)

Figure 24 is another traffic sign with one straight arrow and one left arrow. Table 5 lists the image strings of Figure 24. To distinguish between signs which have identical objects, the system might divide an object into several parts. For example, the arrow here is divided into two parts: arrow and arrow_head. For the reason of flexibility, an object might be stored in the system with several aliases, e.g., a car could be an auto, an automobile, a vehicle, . . . , etc.

10.3 Sample Queries

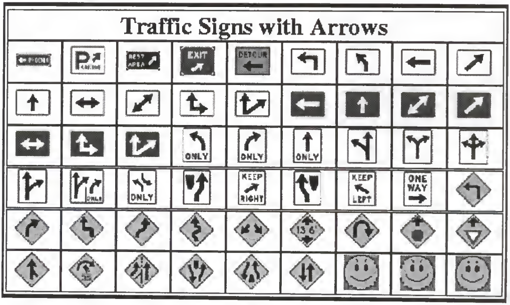


Figure 25. Traffic signs with arrows.

Figure 25 lists fifty-one traffic signs with different kinds of arrows in this system. To retrieve one traffic sign with a specific arrow is the best way to show the strength of this system. Some permutations of the substrings in Table 5 will be used in the following queries to retrieve the image of Figure 24. Figure 26 is a search results list for a query specifying the rectangular system, the neural network search method, and a centroid string: (arrow rectangle, rectangle arrow). The list includes images with arrows and rectangular frames. The results are not satisfied, since a centroid string provides little information for the system to use.

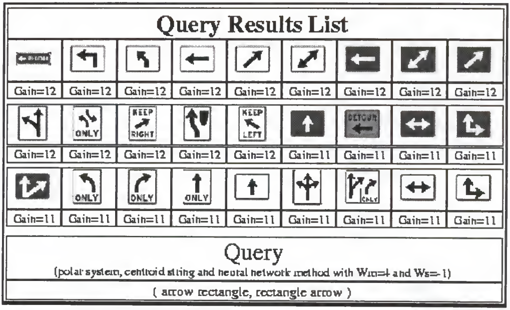


Figure 26. Search results by using a centroid string.

The next two figures demonstrate that the ALACS method outperforms the LCS method. Figure 27 is the search results list for a query specifying the rectangular system, the LCS method, and a line string. Figure 28 is the results list for the same query, shown in Figure 27, except with the ALACS method. Both results show the desired target image, given in Figure 24, is located in the second category of gain. The ALACS method is

shown to outperform the LCS method, since it has fewer elements at that category.

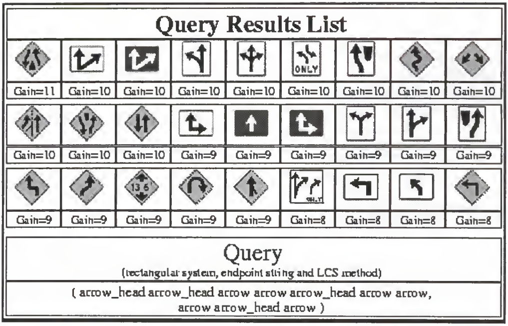


Figure 27. Search results by using the LCS method.

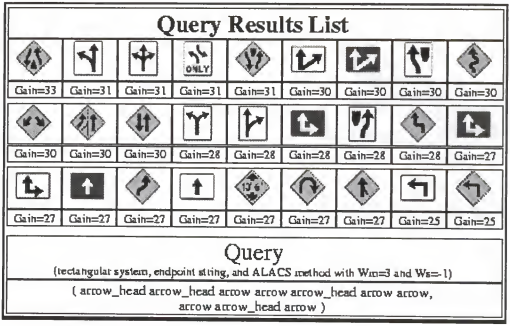


Figure 28. Search results by using the ALACS method.

The last two search results will show the differences between the rectangular and polar coordination systems used in the proposed system. The search results for queries

specifying the rectangular and the polar systems are illustrated in Figure 29 and 30, respectively. Each query string is created from the second half of its respective line

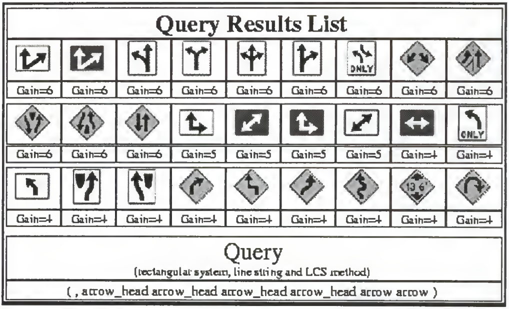


Figure 29. Search results by using the rectangular system.

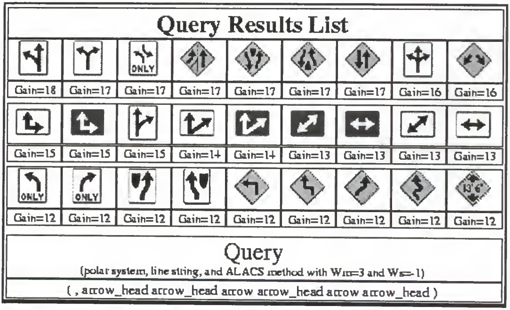


Figure 30. Search results by using the polar system.

string. Since there is no permutation operation on the strings, the search results are not different between the LCS and ALACS methods. The query with the polar system

successfully retrieves the desired traffic sign, but the query with rectangular system does not. Both the length and angle data, given by the polar system, and the x and y coordinates data, given by the rectangular system, supply different and useful information. It is an advantage for the system to provide both coordination systems.

CHAPTER 11
CONCLUSIONS

None of the previous image representations or image matching methods has satisfied fully the two major demands of an image database: (i) accurate and fast retrieval and (ii) efficient storage. This study attempts to meet the above two needs through a line-string image representation and through a new image matching method. The representation better characterizes objects yet preserves an object-oriented property. The technique of finding a longest approximate common subsequence improves the image retrieval methods. However, assuring all the needs of an image database remains an elusive goal. Conclusions of these works and possible future tasks are discussed individually on each research area.

11.1 A Line-String Image Representation

To utilize fully the system's superiority, system users need to be familiar with the image skeletonizing technique. The Zhang-Suen transform repeatedly removes boundary points from a region in a binary image until an irreducible skeleton remains. Some special cases should be noted, e.g., the skeleton of a disk is a dot. The outcome from *APPROX_LINE* algorithm applied on a curve is uncertain. Therefore, images are represented approximately by line strings. This makes the system's performance depend on the strength of the fuzzy matching algorithm. Inferior descriptions of objects' relative positions, forms, and sizes are disadvantages for all string image representations. Chain code [18, 19] and polygonal approximation [44, 40, 3] representations are worthy candidates for alternate image representation approaches.

11.2 A Line-String Matching Method

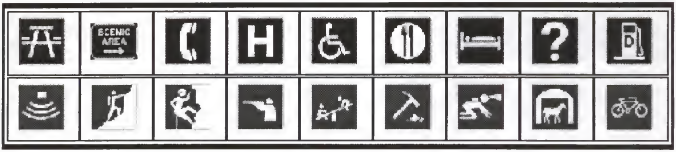
This study is a preliminary look at the LACS approximation problem, but several open questions need answers before it becomes definitive. Thus far, only the last case of $LACS_i(X,Y)$, $1 \leq i \leq \min\{|X|,|Y|\}-1$, has been proven to be NP-hard. The other cases remain open, but it is likely that the vertex-cover problem [28] may be reduced in NP-hard proofs for these cases. For the LACS heuristic approximation algorithm, an enduring difficulty is the worst ratio bound. Either it must prove to have a good average ratio bound, or another good ratio bound algorithm must be designed. Presently, only the worst ratio bound exists for the algorithm, which is $(i+2)/2$ for $ALACS_i$. It is believed that the actual ratio bound is a much smaller number. The performance of LACS Hopfield-style net is slightly worse than the heuristic algorithm. This is due to the problem of local minima. Adding noise terms to the net input of each neuron is being investigated as a way to avoid this problem. The LACS definition does not cover the equal signs in strings. Including equal signs in strings would create at least an NP-hard problem for the matching methods.

11.3 An Image Retrieval System

A traffic sign retrieval system has been developed based on an efficient line-string image data structure and an effective line-string matching method. Preliminary results on the traffic sign image database show that this is a promising technique for content-based image database retrieval. There are a number of research issues which need to be addressed to improve the system performance. Simple images like the traffic signs are the images stored in the system. The proposed image data structure and image matching method are capable of handling complicated images. The problem lays on image segmentation algorithms, which fail on complex images. A future extension may be in the direction of allowing hand-drawn sketch query, which matches objects' sketches.

Edge detection algorithms, which are used to find objects' sketches, are more robust than segmentation algorithms.

A Prototype Traffic Sign Retrieval System



Line String Query

- Polar or Rectangular system
- LCS LACS ALACS or NN search method
  If not LCS, then W_m: ________ and W_s: ________
- Centroid string (X_c, Y_c) or (R_c, A_c):
  ____

  Endpoint string (X_p, Y_p) or (R_p, A_p):
  ____

  Line string (X_l, Y_l) or (R_l, A_l):
  ____

Send Clear Help

APPENDIX
PROOFS OF THEOREMS

Proof of Lemma I: Knuth shows the number of ways to partition a set of n elements into m nonempty disjoint subsets is the Stirling number of the second kind $\left\{{n \atop m}\right\}$.[30] For a set of *NC* centroids with *NE* equal signs, there are *NC-NE* nonempty sets, {centroids separated by '='s}, in a partition. Therefore, the number of ways to partition a set of *NC* centroids with *NE* equal signs into *NC-NE* nonempty sets is $\left\{{NC \atop NC-NE}\right\}$. Each set in a partition is either a single centroid set or an '=' separates every two centroids in that set. For a partition, the number of set orderings is (*NC-NE*)!. Taking into account the number of ways of partition and the number of set orderings, this shows that the number of spatial relations of r_c or θ_c is $\left\{{NC \atop NC-NE}\right\}(NC-NE)!$. □

Proof of Lemma II: For *NO* objects and *NP* endpoints, the multiset of *NP* endpoints can be partitioned into various nonempty endpoint multisets S_h={object h's endpoints}, $1 \le h \le NO$ such that $\biguplus_{h=1}^{NO} S_h = U = multiset\ of\ all\ endpoints$ and $|U|=NP$. One above collection $\biguplus_{h=1}^{NO} S_h$ with *NE* equal signs can be further partitioned into *NP-NE* nonempty multisets, {endpoints separated by '='s}. Consequently, the collection $\biguplus_{h=1}^{NO} S_h$ with *NE* equal signs can be partitioned into nonempty multisets $n_i \times T_i$={endpoints separated by '='s} with $n_i > 0$ and $1 \le i \le I$ for which $\biguplus_{i=1}^{I} T_i^{n_i} = \biguplus_{h=1}^{NO} S_h$ and $\sum_{i=1}^{I} n_i = NP - NE$. Without considering repetitive multisets, the number of these multiset orderings is (*NP-NE*)!. Summing up the two partitionings and the number of orderings, we now have

$\sum_{\biguplus_{h=1}^{NO} S_h=U,|U|=NP} \sum_{\biguplus_{i=1}^{I} T_i^{n_i}=\biguplus_{h=1}^{NO} S_h}$ (*NP-NE*)! spatial relations.

To count out the duplication of these repetitive multisets, the above result should be divided by $\prod_{i=1}^{I} n_i!$ since there are n_i repetitions for each multiset T_i. This shows that:

$$NSR_p'(NO,NP,NE) = \sum_{\biguplus_{h=1}^{NO} S_h=U,|U|=NP} NSR_p''(NO,NP,NE), \ and$$

$$NSR_p''(NO,NP,NE) = \sum_{\biguplus_{i=1}^{I} T_i^{n_i}=\biguplus_{h=1}^{NO} S_h} \frac{(NP-NE)!}{\prod_{i=1}^{I} n_i!}$$

□

Proof of Theorem II: The number of spatial relations of (r_p,θ_p) is equal to the product of numbers of spatial relations of r_p and θ_p minus the illegal formations created from the product. The illegal formation exists when one endpoint of an object overlays another endpoint of the same object, i.e., they have the same coordinates. Therefore, when the statement $(|S_h \cap T_i| > \lceil |S_h|/2 \rceil) \wedge (|S_h \cap R_j| > \lceil |S_h|/2 \rceil)$ is true, then $\sum_{\biguplus_{h=1}^{NO} S_h=U,|U|=NP} ILL_p'(NO,NP,NE_r,NE_\theta) > 0$. $|S_h \cap T_i| > \lceil |S_h|/2 \rceil$ means more than half of object h's endpoints are having the same length; $|S_h \cap R_j| > \lceil |S_h|/2 \rceil$ represents more than half of object h's endpoints are having the same angle; and joining both makes at least two endpoints of object h be superimposing on each other. $\sum_{i=1}^{I} n_i = NP - NE_r$ since there are $NP\text{-}NE_r$ multisets, {endpoints in r_p separated by '='s}, in a partition of $\biguplus_{h=1}^{NO} S_h$, so is $\sum_{j=1}^{J} m_j = NP - NE_\theta$ to θ_p. □

Proof of Lemma III: The proof is similar to the proof of Lemma II except that the line segment multisets T_i, *1≤i≤NO* must conform to the following conditions:

1. T_i={object i's line segments}, $1 \le i \le NO$ for which $0 \le |T_i| \le \binom{|S_i|}{2}$ since the maximum number of connected lines for $|S_i|$ points is $\binom{|S_i|}{2}$,[12] and
2. $\biguplus_{i=1}^{NO} T_i = V = multiset\ of\ all\ line\ segments$ and $|V|=NL$.

In addition, $\sum_{j=1}^{J} m_j = NL - NE$ since there are NL-LE multisets, {line segments separated by '='s}, in a partition of $\biguplus_{i=1}^{NO} T_i$. □

Proof of Theorem III: Similar to Lemma III with Theorem II. □

Proof of Theorem V: This formula just collects of the above theorems. The lower bound of NSR_{ls} is ΣNSR_p or ΣNSR_{pl} when $NC>0$ and $NP>0$. The upper bound of NSR_{ls} is $\Sigma NSR_c \times NSR_{pl}$ or $\Sigma NSR_c \times NSR_p$ because the products may create some bizarre arrangements, e.g., when $NC=2$ and $NP=2$, the number of spatial relations is 4, not 16 because the centroids coincide with the endpoints in this case. In the future, we will try to find the equation of NSR_{ls}, not just its lower or upper bounds. □

REFERENCES

[1] R.A. Baeza-Yates and G.H. Gonnet. A new approach to text searching. *Communications of the ACM*, 35(10):74–82, October 1992.

[2] R.A. Baeza-Yates and C.H. Perleberg. Fast and practical approximate pattern matching. In *Proceedings of Third Conference on Combinatorial Pattern Matching*, pages 182–189, Tucson, Ariz., April 1992.

[3] A. Bengtsson and J.O. Eklundh. Shape representation by multiscale contour approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):85–93, 1991.

[4] S.K. Bhatia and C.L. Sabharwal. A fast implementation of a perfect hash function for picture objects. *Pattern Recognition*, 27(1):365–376, 1974.

[5] R. Boyer and S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, October 1977.

[6] D.W. Bradley and R.A. Bradley. Application of sequence comparison to the study of bird songs. In D. Sankoff and J.B. Kruskal, editors, *In Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 189–214. Addison-Wesley, Reading, MA, 1983.

[7] C.C. Chang and A. Hsu. Image information systems: where do we go from here? *IEEE Transactions on Knowledge and Data Engineering*, 4(5):431–442, October 1992.

[8] C.C. Chang and C.F. Lee. Relative coordinates oriented symbolic string for spatial relationship retrieval. *Pattern Recognition*, 28(4):563–570, 1995.

[9] C.C. Chang and S.Y. Lee. Retrieval of similar pictures on pictorial databases. *Pattern Recognition*, 24(7):675–680, 1991.

[10] S.K. Chang, J.Y. Shi, and C.W. Yan. Iconic indexing by 2-D strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):413–428, May 1987.

[11] S.K. Chang, C.W. Yan, D.C. Dimitroff, and T. Arndt. An intelligent image database system. *IEEE Transactions on Software Engineering*, 14(5):681–688, May 1988.

[12] G. Chartrand and L. Lesniak. *Graphs and digraphs*. Wadsworth and Brooks/Cole, Monterey, CA, second edition, 1986.

[13] C.Y. Chen and C.C. Chang. An object-oriented similarity retrieval algorithm for iconic image databases. *Pattern Recognition Letters*, 14:465–470, June 1993.

[14] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to algorithms*. The MIT Press, Cambridge, MA, 1990.

[15] N.R. Dixon and T.B. Martin, editors. *Automatic speech and speaker recognition*. IEEE Press, New York City, 1979.

[16] C.W. Ferguson. Dendrochronology of bristlecone pine, *pinus aristata*. establishment of a 7484-year chronology in the white mountains of eastern-central california, u.s.a. In I.U. Olsson, editor, *Radiocarbon variations and absolute chronology*, pages 237–259. New York: Wiley Interscience, and Stockholm: Almqvist and Wiksell, 1970.

[17] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *IEEE Computer*, 28(9):23–32, September 1995.

[18] H. Freeman. On the encoding of arbitrary geometric configurations. *IEEE Transactions on Electronic Computers*, EC-10:260–268, 1961.

[19] H. Freeman. Computer processing of line drawings. *Computing Surveys*, 6:57–97, 1974.

[20] P.A.V. Hall and G.R. Dowling. Approximate string matching. *Computing Surveys*, 12(4):381–402, 1980.

[21] D.S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664–675, October 1977.

[22] J.J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. In *Proceedings of the National Academy of Science of the USA*, volume 81, pages 3088–3092, 1984.

[23] A. Hornsten. Summary of discussion of the measurements and identification of varves. In I.U. Olsson, editor, *Radiocarbon variations and absolute chronology*, pages 215–217. New York: Wiley Interscience, and Stockholm: Almqvist and Wiksell, 1970.

[24] P.W. Huang and Y.R. Jean. Using 2d c^+-strings as spatial knowledge representation for image database systems. *Pattern Recognition*, 27(9):1249–1257, 1994.

[25] P.W. Huang and Y.R. Jean. Spatial reasoning and similarity retrieval for image database systems based on rs-strings. *Pattern Recognition*, 29(12):2103–2114, 1996.

[26] A.K. Jain and A. Vailaya. Shape-based retrieval: A case study with trademark image databases. *Pattern Recognition*, 31(9):1369–1390, 1998.

[27] B. Julesz. *Foundations of cyclopean perception.* University of Chicago Press, Chicago, 1971.

[28] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[29] R. Karp and M. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31:249–260, 1987.

[30] D.E. Knuth. *The art of computer programming*, volume I, II, and III. Addison-Wesley, Reading, MA, 1973.

[31] D.E. Knuth, J. Morris, and V. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6:323–350, 1977.

[32] J. B. Kruskal. An overview of sequence comparison. In D. Sankoff and J.B. Kruskal, editors, *In Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 1–44. Addison-Wesley, Reading, MA, 1983.

[33] S.Y. Lee and F.J. Hsu. 2D C-string: A new spatial knowledge representation for image database systems. *Pattern Recognition*, 23(10):1077–1087, 1990.

[34] S.Y. Lee and F.J. Hsu. Spatial reasoning and similarity of images using 2d c-string knowledge representation. *Pattern Recognition*, 25(3):305–318, 1992.

[35] S.Y. Lee and M.K. Shan. Access methods of image database. *International Journal of Pattern Recognition and Artificial Intelligence*, 4(1):27–44, 1990.

[36] S.Y. Lee, M.K. Shan, and W.P. Yang. Similarity retrieval of iconic image database. *Pattern Recognition*, 22(6):675–682, 1989.

[37] V.I. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1(1):8–17, 1965.

[38] R. Lowrance and R.A. Wagner. An extension of the string-to-string correction problem. *Journal of the ACM*, 22(2):177–183, 1975.

[39] W.J. Masek and M.S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20:18–31, 1980.

[40] T. Pavlidis. *Structural pattern recognition.* Spring-Verlag, New York, 1977.

[41] A. Pentland, R.W. Picard, and S. Sclaroff. Photobook: content-based manipulation of image databases. *International Journal of Computer Vision*, 18(3):233–254, June 1996.

[42] C.L. Sabharwal and S.K. Bhatia. Perfect hash table algorithm for image databases using negative associated values. *Pattern Recognition*, 28(7):1091–1101, 1995.

[43] C.L. Sabharwal and S.K. Bhatia. Image databases and near-perfect hash table. *Pattern Recognition*, 30(11):1867–1876, 1997.

[44] J. Sklansky, R.L. Chazin, and B.J. Hansen. Minimum-perimeter polygons of digitized silhouettes. *IEEE Transactions on Computers*, C-21(3):260–268, 1972.

[45] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[46] Y. Fujimoto *et al.*. Recognition of handprinted characters by nonlinear elastic matching. In *3rd International Joint Conference on Pattern Recognition*, pages 113–119, Coronado, CA, 1976. IEEE.

[47] E. Ukkonen. Finding approximate patterns in strings. *Journal of Algorithms*, 6:132–137, 1985.

[48] R.A. Wagner. On the complexity of the extended string-to-string correction problem. *Proc. Seventh Annual ACM Symp. on Theory of Computing*, pages 218–223, 1975.

[49] R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, January 1974.

[50] W.J. Wilbur and D.J. Lipman. Rapid similarity searches of nucleic acid and protein data banks. In *Proceedings of the National Academy of Science of the USA*, volume 80, pages 726–730, 1983.

[51] S. Wu and U. Manber. Text searching allowing errors. *Communications of the ACM*, 35(10):83–91, October 1992.

[52] T.C. Wu and C.C. Chang. Application of geometric hashing to iconic database retrieval. *Pattern Recognition Letters*, 15:871–876, September 1994.

[53] T.Y. Zhang and C.Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.

BIOGRAPHICAL SKETCH

Wen-Chen Hu was born on 31 October, 1960 in Taipei, Taiwan. He received a B.S. degree in computer science from Tamkang University, Taipei, Taiwan in 1984, a M.S. degree in electronic and information engineering from National Central University, Chungli, Taiwan in 1986, and a M.S. degree in computer science from the University of Iowa, Iowa City in 1993. Now, he will receive his Doctor of Philosophy degree in computer and information science and engineering from the University of Florida, Gainesville, Florida in August 1998. His current research interests include multimedia databases, computer vision, and programming languages.